

Network Working Group
INTERNET-DRAFT
Intended Category: Standards Track
June 7, 2004
Expires December 6, 2004

Rob Weltman
Netscape Communications Corp.
Christine Tomlinson
Sun Microsystems, Inc.
Steven Sonntag
Novell, Inc.

The Java LDAP Application Program Interface
[draft-ietf-ldapext-ldap-java-api-19.txt](#)

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with [RFC 3668](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on November 22, 2004.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This document defines a Java [[JAVA](#)] language application program interface to the Lightweight Directory Access Protocol Version 3 (LDAP) [[LDAPv3](#)], in the form of a class library.

Conventions Used in this Document

The key words "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", and "MAY"

Expires December 6, 2004

[Page 1]

in this document are to be interpreted as defined in "Key words for use in RFCs to Indicate Requirement Levels" [[KEYWORDS](#)].

Expires December 6, 2004

[Page 2]

1. Overview.....	10
1.1 The LDAP model.....	10
1.2 Package name.....	11
1.3 The LDAP classes.....	11
1.4 The LDAP asynchronous methods.....	12
1.5 Interfaces.....	12
1.6 Classes.....	12
1.7 Exceptions.....	15
1.8 LDAP API use.....	15
2. The Java LDAP classes.....	17
2.1 public class LDAPAttribute.....	18
2.1.1 Constructors.....	18
2.1.2 addValue.....	19
2.1.3 compareTo.....	19
2.1.4 getBaseName.....	19
2.1.5 getByteValues.....	20
2.1.6 getByteValueArray.....	20
2.1.7 getLangSubtype.....	20
2.1.8 getName.....	20
2.1.9 getStringValueArray.....	20
2.1.10 getStringValues.....	20
2.1.11 getSubtypes.....	21
2.1.12 hasSubtype.....	21
2.1.13 hasSubtypes.....	21
2.1.14 removeValue.....	21
2.1.15 size.....	22
2.2 public class LDAPAttributeSchema.....	22
2.2.1 Constructors.....	22
2.2.2 getEqualityMatchingRule.....	23
2.2.3 getOrderingMatchingRule.....	24
2.2.4 getSubstringMatchingRule.....	24
2.2.5 getSuperior.....	24
2.2.6 getSyntaxString.....	24
2.2.7 getUsage.....	24
2.2.8 isCollective.....	24
2.2.9 isSingleValued.....	25
2.2.10 isUserModifiable.....	25
2.2.11 Constants of LDAPAttributeSchema.....	25
2.3 public class LDAPAttributeSet.....	25
2.3.1 Constructors.....	26
2.3.2 clone.....	26
2.3.3 getAttribute.....	26
2.3.4 getSubset.....	26
2.4 public interface LDAPAuthHandler.....	28
2.4.1 getAuthProvider.....	28
2.5 public class LDAPAuthProvider.....	28
2.5.1 Constructors.....	28
2.5.2 getDN.....	29

2.5.3	getPassword.....	29
2.6	public interface LDAPBindHandler.....	29
2.6.1	bind.....	29
2.7	public class LDAPCompareAttrNames.....	30

2.7.1	Constructors	30
2.7.2	compare	31
2.7.3	equals	31
2.7.4	getLocale	32
2.7.5	setLocale	32
2.8	public class LDAPConnection	32
2.8.1	Constructors	32
2.8.2	abandon	33
2.8.3	add	34
2.8.4	addUnsolicitedNotificationListener	34
2.8.5	bind	35
2.8.6	clone	38
2.8.7	compare	39
2.8.8	connect	39
2.8.9	delete	40
2.8.10	disconnect	41
2.8.11	extendedOperation	41
2.8.12	fetchSchema	42
2.8.13	finalize	42
2.8.14	getAuthenticationDN	42
2.8.15	getAuthenticationMethod	43
2.8.16	getConstraints	43
2.8.17	getHost	43
2.8.18	getPort	43
2.8.19	getProperty	44
2.8.20	getProtocolVersion	44
2.8.21	getResponseControls	45
2.8.22	getSaslBindCallbackHandler	45
2.8.23	getSaslBindProperties	45
2.8.24	getSchemaDN	45
2.8.25	getSearchConstraints	46
2.8.26	getSocketFactory	46
2.8.27	isBound	46
2.8.28	isConnected	46
2.8.29	isTLS	46
2.8.30	modify	46
2.8.31	read	48
2.8.32	removeUnsolicitedNotificationListener	49
2.8.33	rename	50
2.8.34	search	51
2.8.35	setConstraints	54
2.8.36	setSocketFactory	54
2.8.37	startTLS	54
2.8.38	stopTLS	55
2.8.39	Constants of LDAPConnection	55
2.9	public class LDAPConstraints	56
2.9.1	Constructors	56
2.9.2	getControls	57

2.9.3	getHopLimit	57
2.9.4	getProperty	57
2.9.5	getReferralFollowing	57
2.9.6	getTimeLimit	57

Expires December 6, 2004

[Page 4]

2.9.7	setControls	58
2.9.8	setHopLimit	58
2.9.9	setProperty	58
2.9.10	setReferralFollowing	59
2.9.11	setReferralHandler	59
2.9.12	setTimeLimit	59
2.10	public class LDAPControl	59
2.10.1	Constructors	60
2.10.2	clone	60
2.10.3	getID	60
2.10.4	getValue	60
2.10.5	isCritical	60
2.10.6	register	61
2.10.7	setValue	61
2.11	public class LDAPDITContentRuleSchema	61
2.11.1	Constructors	61
2.11.2	getAuxiliaryClasses	62
2.11.3	getOptionalAttributes	62
2.11.4	getPrecludedAttributes	63
2.11.5	getRequiredAttributes	63
2.12	public class LDAPDITStructureRuleSchema	63
2.12.1	Constructors	63
2.12.2	getNameForm	64
2.12.3	getRuleID	64
2.12.4	getSuperiors	64
2.13	public class LDAPDN	65
2.13.1	equals	65
2.13.2	escapeRDN	65
2.13.3	explodeDN	65
2.13.4	explodeRDN	66
2.13.5	isValid	66
2.13.6	normalize	66
2.13.7	unescapeRDN	66
2.14	public class LDAPEntry	67
2.14.1	Constructors	67
2.14.2	compareTo	67
2.14.3	getAttribute	68
2.14.4	getAttributeSet	68
2.14.5	getDN	68
2.15	public class LDAPException	69
2.15.1	Constructors	69
2.15.2	getCause	70
2.15.3	getLDAPErrorMessage	70
2.15.4	getResultCode	71
2.15.5	getMatchedDN	71
2.15.6	resultCodeToString	71
2.15.7	toString	72
2.15.8	Result codes	72

2.16	public class LDAPExtendedOperation.....	73
2.16.1	Constructors.....	73
2.16.2	getID.....	74
2.16.3	getValue.....	74

Expires December 6, 2004

[Page 5]

2.16.4	setValue.....	74
2.17	public class LDAPExtendedResponse.....	74
2.17.1	getID.....	74
2.17.2	getValue.....	74
2.17.3	register.....	75
2.18	public class LDAPLocalException.....	75
2.18.1	Constructors.....	75
2.19	public class LDAPMatchingRuleSchema.....	76
2.19.1	Constructors.....	76
2.19.2	getAttributes.....	77
2.19.3	getSyntaxString.....	77
2.20	public class LDAPMatchingRuleUseSchema.....	77
2.20.1	Constructors.....	77
2.20.2	getAttributes.....	78
2.21	public class LDAPMessage.....	78
2.21.1	getControls.....	78
2.21.2	getMessageID.....	79
2.21.3	getType.....	79
2.22	public interface LDAPMessageQueue.....	79
2.22.1	getMessageIDs.....	79
2.22.2	getResponse.....	80
2.22.3	isResponseReceived.....	80
2.22.4	merge.....	81
2.23	public class LDAPModification.....	81
2.23.1	Constructors.....	81
2.23.2	getAttribute.....	82
2.23.3	getOp.....	82
2.23.4	Constants of LDAPModification.....	82
2.24	public class LDAPNameFormSchema.....	82
2.24.1	Constructors.....	83
2.24.2	getObjectClass.....	83
2.24.3	getOptionalNamingAttributes.....	84
2.24.4	getRequiredNamingAttributes.....	84
2.25	public class LDAPObjectClassSchema.....	84
2.25.1	Constructors.....	84
2.25.2	getOptionalAttributes.....	85
2.25.3	getRequiredAttributes.....	85
2.25.4	getSuperiors.....	85
2.25.5	getType.....	85
2.25.6	Constants of LDAPObjectClassSchema.....	86
2.26	public class LDAPReferralException.....	86
2.26.1	Constructors.....	86
2.26.2	getFailedReferral.....	87
2.26.3	getReferrals.....	87
2.26.4	setFailedReferral.....	87
2.27	public interface LDAPReferralHandler.....	87
2.28	public class LDAPResponse.....	87
2.28.1	getErrorMessage.....	88

2.28.2	getMatchedDN.....	88
2.28.3	getReferrals.....	88
2.28.4	getResultCode.....	88
2.29	public class LDAPResponseQueue.....	88

2.29.1	getMessageIDs.....	88
2.29.2	getResponse.....	88
2.29.3	isResponseReceived.....	88
2.29.4	merge.....	88
2.30	public class LDAPSchema.....	88
2.30.1	Constructors.....	88
2.30.2	getAttributeNames.....	89
2.30.3	getAttributeSchema.....	89
2.30.4	getAttributeSchemas.....	89
2.30.5	getDITContentRuleNames.....	89
2.30.6	getDITContentRuleSchema.....	89
2.30.7	getDITContentRuleSchemas.....	90
2.30.8	getDITStructureRuleNames.....	90
2.30.9	getDITStructureRuleSchema.....	90
2.30.10	getDITStructureRuleSchemas.....	90
2.30.11	getMatchingRuleNames.....	90
2.30.12	getMatchingRuleSchema.....	90
2.30.13	getMatchingRuleSchemas.....	91
2.30.14	getMatchingRuleUseNames.....	91
2.30.15	getMatchingRuleUseSchema.....	91
2.30.16	getMatchingRuleUseSchemas.....	91
2.30.17	getNameFormNames.....	91
2.30.18	getNameFormSchema.....	91
2.30.19	getNameFormSchemas.....	92
2.30.20	getObjectClassNames.....	92
2.30.21	getObjectClassSchema.....	92
2.30.22	getObjectClassSchemas.....	92
2.30.23	getSyntaxSchema.....	92
2.30.24	getSyntaxSchemas.....	93
2.31	public abstract class LDAPSchemaElement.....	93
2.31.1	getDescription.....	93
2.31.2	getNames.....	93
2.31.3	getID.....	93
2.31.4	getQualifier.....	93
2.31.5	getQualifierNames.....	94
2.31.6	isObsolete.....	94
2.31.7	setQualifier.....	94
2.31.8	toString.....	94
2.32	public class LDAPSearchConstraints.....	94
2.32.1	Constructors.....	95
2.32.2	getBatchSize.....	96
2.32.3	getDereference.....	96
2.32.4	getMaxResults.....	96
2.32.5	getServerTimeLimit.....	97
2.32.6	setBatchSize.....	97
2.32.7	setDereference.....	97
2.32.8	setMaxResults.....	97
2.32.9	setServerTimeLimit.....	98

2.32.10	Constants of LDAPSearchConstraints.....	98
2.33	public class LDAPSearchQueue.....	98
2.33.1	getMessageIDs.....	98
2.33.2	getResponse.....	98

2.33.3	isComplete	98
2.33.4	isResponseReceived	98
2.33.5	merge	98
2.34	public class LDAPSearchResult	98
2.34.1	getEntry	98
2.35	public class LDAPSearchResultReference	98
2.35.1	getReferrals	98
2.36	public class LDAPSearchResults	99
2.36.1	getCount	99
2.36.2	getResponseControls	99
2.36.3	hasMore	99
2.36.4	next	99
2.37	public interface LDAPSocketFactory	99
2.37.1	createSocket	99
2.38	public class LDAPSyntaxSchema	99
2.38.1	Constructors	100
2.39	public interface LDAPTLSSocketFactory	100
2.39.1	createSocket	100
2.40	public interface LDAPUnsolicitedNotificationListener	100
2.40.1	messageReceived	100
2.41	public class LDAPUrl	101
2.41.1	Constructors	101
2.41.2	decode	102
2.41.3	encode	102
2.41.4	getAttributeArray	102
2.41.5	getAttributes	103
2.41.6	getDN	103
2.41.7	getExtensions	103
2.41.8	getFilter	103
2.41.9	getHost	103
2.41.10	getPort	103
2.41.11	getScope	103
2.41.12	toString	104
3	Implementation considerations	104
3.1	Controls	104
3.2	Referral handling and exceptions	104
3.3	Message IDs	106
3.4	Notice of disconnection	106
3.5	Level of compatibility	107
3.6	Dependencies	107
3.7	Invalid responses	107
4	Security considerations	108
5	Acknowledgements	109
6	Bibliography	109
6.1	Normative References	109
6.2	Informative References	110
7	Authors' addresses	110
8	Appendix A - Sample Java LDAP programs	112

8.1	Java LDAP programs using synchronous methods.....	112
8.2	Java LDAP programs using asynchronous methods.....	118
9	Appendix B - Revision history.....	124
9.1	Changes from ldap-java-api-17.txt.....	126

Expires December 6, 2004

[Page 8]

9.2	Changes from ldap-java-api-16.txt.....	128
9.3	Changes from ldap-java-api-15.txt.....	128
9.4	Changes from ldap-java-api-14.txt.....	132
9.5	Changes from ldap-java-api-13.txt.....	133
9.6	Changes from ldap-java-api-12.txt.....	134
9.7	Changes from ldap-java-api-11.txt.....	136
9.8	Changes from ldap-java-api-10.txt.....	138
9.9	Changes from ldap-java-api-09.txt.....	139
9.10	Changes from ldap-java-api-08.txt.....	140
9.11	Changes from ldap-java-api-07.txt.....	140
9.12	Changes from ldap-java-api-06.txt.....	141
9.13	Changes from ldap-java-api-05.txt.....	142
9.14	Changes from ldap-java-api-04.txt.....	142
9.15	Changes from ldap-java-api-03.txt.....	143
9.16	Changes from ldap-java-api-02.txt.....	144
9.17	Changes from ldap-java-api-01.txt.....	144

Expires December 6, 2004

[Page 9]

1. Overview

The LDAP [[LDAPv3](#)] class library is designed to provide powerful, yet simple, access to LDAP directory services. It defines both asynchronous and synchronous APIs to LDAP that suit a wide variety of client applications and is capable of generating all possible protocol requests and interpreting all possible protocol responses as defined by LDAP v3 [[LDAPPROTO](#)]. This API does not attempt to provide compatibility with earlier versions of LDAP.

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.

1.1 The LDAP model

LDAP is the Lightweight Directory Access Protocol, described in [[LDAPv3](#)]. It defines a lightweight access mechanism by which client applications send requests to and receive responses from LDAP servers.

The LDAP information model comes from X.500 [[X500](#)] 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 directory operations act upon these values.

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 cn 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 Example company, the DN of her entry might be "cn=Barbara Jensen,dc=example,dc=com". The DN format used by LDAP is defined in [[DN](#)].

Objects in LDAP are identified by an Object Identifier in dot-decimal format. Short names are often used as more readable aliases for

Object Identifiers. Object Identifiers in dot-decimal format will be referred to as an OID or as OIDs throughout this document.

Operations are provided to authenticate, search for and retrieve information, modify information, and add and delete entries from the tree. The protocol is also extensible, allowing operations to be extended by "controls" and new "extended" operations to be defined.

An LDAP server may return referrals or search references 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 server may return a referral. If a search request spans multiple servers, it may return one or more search references).

1.2 Limitations

Before the API implementation encodes and sends a string value to a server, the string values are converted from the Java 16-bit Unicode format (UCS2) to UTF-8 format, which many LDAPv3 protocol elements and valueencodings use. The integrity of double-byte and other non-ASCII character sets is fully preserved. Any characters to be sent or received, which cannot be represented with Java 16-bit Unicode strings must be processed as binary values by the client application. Values received from a server which cannot be represented as UCS-2 characters must be handled as binary values, since they will produce undefined results if converted to a Java String.

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.

1.3 Package name

The classes of the LDAP class library have the package name `org.ietf.ldap`.

1.4 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, delete entries in the directory, and establish integrity and confidentiality protective services.

The `LDAPConnection` class also provides access to 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 `LDAPConstraints` or `LDAPSearchConstraints`).

A synchronous 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.

1.5 The LDAP asynchronous methods

The LDAP protocol provides synchronous as well as asynchronous directory access methods. All asynchronous methods are conducted by an LDAPConnection object, take an LDAPMessageQueue object as input, and return an LDAPMessageQueue object. The returned LDAPMessageQueue object is a message queue associated with the request, and it is the responsibility of the client application to read messages out of the queue and process them.

Messages retrieved from an LDAPMessageQueue are objects of type LDAPResponse, LDAPSearchResult, or LDAPSearchResultReference. .

An asynchronous search returns an LDAPMessageQueue object. Search results are obtained from that object via the getResponse method. A search result is typically an LDAPSearchResult object, which has a getEntry method. The LDAPEntry returned by getEntry contains the DN and attributes of a single search result.

None of the ancillary asynchronous classes are intended to be instantiated by a client application, so they lack public constructors.

1.6 Interfaces

LDAPAuthHandler Interface used to provide credentials for simple bind when following a referral.

LDAPBindHandler Interface used to do explicit bind processing when following a referral.

LDAPReferralHandler Interface that is a shared ancestor to LDAPBindHandler and LDAPAuthHandler.

LDAPUnsolicitedNotificationListener Interface that allows a client application to be notified when unsolicited messages arrive from a server.

[1.7](#) **Classes**

Expires December 6, 2004

[Page 12]

LDAPAttribute	Represents the name and values of one attribute of a directory entry.
LDAPAttributeSchema	Represents a definition of an attribute in a Directory Server's subschema.
LDAPAttributeSet	Represents a collection of LDAPAttributes.
LDAPAuthProvider	An encapsulation of reauthentication credentials, used when automatically following referrals.
LDAPCompareAttrNames	An implementation of Comparator to support sorting of search results by one or more attributes.
LDAPConnection	The central point for operations on an LDAP Directory Server.
LDAPConstraints	Defines options controlling all operations on a Directory Server.
LDAPControl	Encapsulates additional parameters for an LDAP operation, sent to or received from a server.
LDAPDITContentRuleSchema	Represents a DIT content rule in a Directory Server's subschema.
LDAPDITStructureRuleSchema	Represents a DIT structure rule in a Directory Server's subschema.
LDAPDN	A utility class to facilitate composition and decomposition of distinguished names (DNs).
LDAPEntry	Represents a single entry in a directory.

Expires December 6, 2004

[Page 13]

LDAPExtendedOperation	Encapsulates the OID and data associated with the sending or receiving of an extended operation.
LDAPExtendedResponse	The response returned by an LDAP server on an extended operation request. It extends LDAPResponse.
LDAPMatchingRuleSchema	Represents the schematic definition of a matching rule in a Directory Server's subschema.
LDAPMatchingRuleUseSchema	Represents a matching rule use in a Directory Server's subschema.
LDAPMessage	Base class for LDAP request and response messages. Subclassed by response messages used in asynchronous operations.
LDAPMessageQueue	Represents a queue of incoming asynchronous messages from the server.
LDAPModification	A single add/delete/replace operation to an LDAPAttribute.
LDAPNameFormSchema	Represents a name form in a Directory Server's subschema.
LDAPObjectClassSchema	Represents the schematic definition of an object class in a Directory Server's subschema.
LDAPResponse	Represents a message received from an LDAP server in response to an asynchronous request. It extends LDAPMessage.
LDAPSchema	Represents the subschema controlling one or more entries held by a Directory Server.

LDAPSchemaElement

Base class for representing LDAP
subschema elements.

Expires December 6, 2004

[Page 14]

LDAPSyntaxSchema	Represents a syntax definition in a Directory Server's subschema.
LDAPSearchConstraints	Defines the options controlling search operations.
LDAPSearchResult	A single search result that is in response to an asynchronous search operation. It extends LDAPMessage.
LDAPSearchResultReference	A continuation reference from an asynchronous search operation. It extends LDAPMessage.
LDAPSearchResults	The enumerable results of a search operation.
LDAPUrl	Represents an LDAP Url [LDAPURL].

[1.8](#) Exceptions

LDAPException	General exception, which includes an error message and an LDAP API local error code or server result code.
LDAPLocalException	Derived from LDAPException and is an exception generated by the API implementation, i.e., an exception not received from the server.
LDAPReferralException	Derived from LDAPException and contains a list of URLs corresponding to a single referral or search continuation response received on an LDAP operation.

[1.9](#) LDAP API use

An application generally uses the LDAP API in four steps.

- Construct an LDAPConnection. Initialize an LDAP session with a

Directory Server. Supplying an optional `SocketFactory` during connection creation may enable an SSL or TLS session. The `LDAPConnection.connect()` call establishes a handle to the session, allowing multiple sessions to be open at once, on different instances of `LDAPConnection`.

- Optionally authenticate to the LDAP server with `LDAPConnection.bind()`.
- Perform some LDAP operations and obtain some results. The synchronous version of `LDAPConnection.search()` returns an `LDAPSearchResults` object which can be enumerated to access all entries found. The asynchronous version of `LDAPConnection.search()` returns an `LDAPMessageQueue`, which is used to read the results of the search. `LDAPConnection.read()` returns a single entry. Other methods allow other operations such as add, delete, and modify to be performed.
- Close the connection. The `LDAPConnection.disconnect()` call closes the connection.

There are both synchronous and asynchronous versions of the LDAP protocol operations described in this specification. Synchronous methods do not return until the operation has completed.

Asynchronous methods take an `LDAPMessageQueue` parameter and return an `LDAPMessageQueue` object which is used to enumerate the responses from the server. A loop is typically used to read from the queue object, which blocks until there is a response available, until the operation has completed.

An `LDAPMessageQueue` may be shared between operations for multiplexing the results. In this case, the object returned on one operation is passed in to one or more other operations, rather than passing in null.

For the asynchronous methods, exceptions are raised only for connection errors and API errors (`LDAPLocalException`). LDAP result messages are converted into `LDAPResponse` objects which are to be checked by the client application for errors and referrals, whereas the synchronous methods throw an `LDAPException` on result codes other than `success(0)`, `compareTrue(5)`, and `compareFalse(6)`.

To facilitate user feedback during synchronous searches, intermediate search results can be obtained before the entire search operation is completed by specifying, in an `LDAPSearchConstraints` object, the number of entries to return at a time.

Errors result in the throwing of an `LDAPException`, with a specific

result code and context-specific textual information, if available.

Expires December 6, 2004

[Page 16]

Methods implemented by the API that return an array MUST return an empty array if no values are present to return, unless otherwise specified.

If null is passed as the value of an LDAPConstraints or LDAPSearchConstraints parameter to an operation, the default constraints are used for that operation.

If null is passed as the value of a DN to an operation it is treated as if it was the empty string.

When using synchronous APIs, the client application doesn't distinguish between LDAP search continuation references and LDAP referrals, as the API presents a unified interface for handling the two. This document generically refers to continuation references and referrals as simply referrals or referral following. The API gives the application two options for handling referrals.

1.9.1 Default Referral Handling

By default, referrals are not followed automatically. The application receives a referral and either ignores it or explicitly issues a new request to the referred-to servers.

1.9.2 Automatic Referral Following

The application, if using synchronous requests, can choose to let the library automatically follow the referrals. When automatic referral following is selected, a referral is followed by default with anonymous credentials using the protocol version, socket factory, and TLS [[TLS](#)][LDAPTLS] of the original connection. Socket factories supplied by the client application can determine if and when TLS client credentials are to be disclosed.

If default referral following is not desired when automatically following referrals, the application can instruct the library to follow referrals with an authenticated connection by providing a reauthentication object to supply credentials for a simple bind.

For greater flexibility, the client application can provide an object that creates, binds, and manages authenticated connections for use by the API implementation when automatically following referrals.

2. The Java LDAP classes

The following sections describe the LDAP classes in more detail.

Expires December 6, 2004

[Page 17]

2.1 public class LDAPAttribute **implements Cloneable, Serializable, Comparable**

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.

It should be noted that attribute name (called Attribute Description in the LDAP Protocol [[LDAPPROTO](#)]) consists of an Attribute Type and Attribute Options. Attribute Type can be expressed as an OID or as one of its short names. The API implementation is not required to make a mapping of short names and the OID. The Attribute Type MAY be followed by one or more options. The implementation MUST treat the name and options as case insensitive and return name and options as lower case strings. No ordering can be implied on the options.

2.1.1 Constructors

```
public LDAPAttribute(LDAPAttribute attr)
```

Constructs an attribute with copies of all values of the input attribute.

```
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:

attr An attribute to use as template.

Expires December 6, 2004

[Page 18]

<code>attrName</code>	Name of the attribute.
<code>attrBytes</code>	Value of the attribute as raw bytes.
<code>attrString</code>	Value of the attribute as a String.
<code>attrStrings</code>	Array of values as Strings.

`IllegalArgumentException` is thrown if any of the attribute values is null.

[2.1.2](#) **addValue**

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

Adds a string value to the attribute.

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

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

Parameters are:

<code>attrString</code>	Value of the attribute as a String.
<code>attrBytes</code>	Value of the attribute as raw bytes.

Adding a value which is already present has no effect.

[2.1.3](#) **compareTo**

```
public int compareTo(Object obj)
```

Compares this object with the specified object for order. Ordering is determined by comparing normalized attribute names and options (see `getName()`) using the `compareTo()` method of the String class. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters are:

<code>obj</code>	The object to be compared to this object.
------------------	---

[2.1.4](#) **getTypeName**

```
public String getTypeName()
```

Expires December 6, 2004

[Page 19]

```
public static String getTypeName(String attrName)
```

Returns the type name of the attribute. For example, if the attribute name is cn;lang-ja;phonetic, this method returns cn. The name may be an OID.

attrName	Name of the attribute to extract the type name from.
----------	--

[2.1.5](#) **getBytesValues**

```
public Enumeration getBytesValues()
```

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

[2.1.6](#) **getBytesValueArray**

```
public byte[][] getBytesValueArray()
```

Returns the values of the attribute as an array of byte[].

[2.1.7](#) **getName**

```
public String getName()
```

Returns the normalized name of the attribute, i.e. the attribute type, and its options, if any.

[2.1.8](#) **getStringValueArray**

```
public String[] getStringValueArray()
```

Returns the values of the attribute as an array of Strings. This method should only be called if the attribute values are known to be strings. The returned Strings have undefined values if the attribute values do not consist of valid UTF-8 character encodings.

[2.1.9](#) **getStringValues**

```
public Enumeration getStringValues()
```

Returns an enumerator for the string values of an attribute. This method should only be called if the attribute values are known to be strings. The returned Stringvalues are undefined if the values do not

consist of valid UTF-8 character encodings.

Expires December 6, 2004

[Page 20]

[2.1.10](#) **getOptions**

```
public String[] getOptions()
```

```
public static String[] getOptions(String attrName)
```

Extracts the options from the specified attribute name. For example, if the attribute name is cn;lang-ja;phonetic, this method returns an array containing lang-ja and phonetic. The options may be returned in any order.

Parameters are:

attrName	Name of the attribute to extract the options from.
----------	--

[2.1.11](#) **hasOption**

```
public boolean hasOption(String option)
```

Reports if the attribute name contains the specified option. For example, if you check for the option lang-en and the attribute name is cn;lang-en;phonetic, this method returns true.

Parameters are:

option	The single option to check for.
--------	---------------------------------

[2.1.12](#) **hasOptions**

```
public boolean hasOptions(String[] options)
```

Reports if the attribute name contains at least the specified options. For example, if you check for the options lang-en and phonetic and if the attribute name is cn;lang-en;phonetic, this method returns true. If the attribute name is cn;phonetic or cn;lang-en, this method returns false. The options may be specified in any order.

Parameters are:

options	An array of subtypes to check for.
---------	------------------------------------

[2.1.13](#) **removeValue**

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

Removes a string value from the attribute.

Expires December 6, 2004

[Page 21]

```
public void removeValue(byte[] attrBytes)
```

Removes a byte[]-formatted value from the attribute. The value to be removed must match, byte for byte, the specified value.

Parameters are:

attrString Value of the attribute as a String.

attrBytes Value of the attribute as raw bytes.

Removing a value which is not present in the attribute has no effect.

[2.1.14](#) size

```
public int size()
```

Returns the number of values of the attribute.

[2.2](#) **public class LDAPAttributeSchema** **extends LDAPSchemaElement**

The LDAPAttributeSchema class represents the definition of an attribute. It is used to query attribute syntax, and to add or delete an attribute definition in a Directory's subschema. See [\[ATTR\]](#) for a description of attribute representation in LDAP.

[2.2.1](#) Constructors

```
public LDAPAttributeSchema(String[] names,  
                           String oid,  
                           String description,  
                           String syntaxString,  
                           boolean single,  
                           String superior,  
                           boolean obsolete,  
                           String equality,  
                           String ordering,  
                           String substring,  
                           boolean collective,  
                           boolean userMod,  
                           int usage)
```

Constructs an attribute definition for adding to or deleting from a

DirectoryÆs subschema.

```
public LDAPAttributeSchema(String raw)
```

Expires December 6, 2004

[Page 22]

Constructs an attribute definition from an encoding using the AttributeTypeDescription syntax [[ATTR](#)].

Parameters are:

names	Name(s) of the attribute.
oid	OID of the attribute.
description	Optional description of the attribute.
syntaxString	OID of the syntax of the attribute.
single	true if the attribute is to be single-valued.
superior	Optional name of the attribute type which this attribute type derives from; null if there is no superior attribute type.
obsolete	true if this attribute is obsolete.
equality	OID of the equality matching rule for the attribute ; or null if none.
ordering	OID of the ordering matching rule for the, or null if none.
substring	OID of the substring matching rule for the attribute, or null if none.
collective	true if this is a collective attribute.
userMod	true if the attribute is modifiable by users.
usage	One of the following constants (see 2.2.11): USER_APPLICATIONS DIRECTORY_OPERATION DISTRIBUTED_OPERATION DSA_OPERATION
raw	An attribute definition encoded using the AttributeTypeDescription syntax [ATTR].

[2.2.2](#) getEqualityMatchingRule

```
public String getEqualityMatchingRule ()
```

Expires December 6, 2004

[Page 23]

Returns the OID of the equality matching rule in effect for this attribute, or null if there is none.

[2.2.3](#) **getOrderingMatchingRule**

```
public String getOrderingMatchingRule ()
```

Returns the OID of the ordering matching rule in effect for this attribute, or null if there is none.

[2.2.4](#) **getSubstringMatchingRule**

```
public String getSubstringMatchingRule ()
```

Returns the OID of the substring matching rule in effect for this attribute, or null if there is none.

[2.2.5](#) **getSuperior**

```
public String getSuperior()
```

Returns the name of the attribute type which this attribute derives from, or null if there is no superior attribute.

[2.2.6](#) **getSyntaxString**

```
public String getSyntaxString()
```

Returns the OID of the syntax of the attribute.

[2.2.7](#) **getUsage**

```
public int getUsage ()
```

Returns one of the following constants (see 2.2.11):

```
USER_APPLICATIONS  
DIRECTORY_OPERATION  
DISTRIBUTED_OPERATION  
DSA_OPERATION
```

[2.2.8](#) **isCollective**

```
public boolean isCollective ()
```

Returns true if the attribute is collective.

Expires December 6, 2004

[Page 24]

[2.2.9](#) **isSingleValued**

```
public boolean isSingleValued()
```

Returns true if the attribute is single-valued.

[2.2.10](#) **isUserModifiable**

```
public boolean isUserModifiable ()
```

Returns true if the attribute is modifiable by users.

[2.2.11](#) **Constants of LDAPAttributeSchema**

The constants correspond to those defined in [RFC 2252](#) [ATTR]: userApplications, directoryOperation, distributedOperation, and dSAOperation. The table below gives the constant name followed by its value.

USER_APPLICATIONS (0)	An ordinary user attribute
DIRECTORY_OPERATION (1)	An operational attribute used for a directory operation or which holds a directory specific value
DISTRIBUTED_OPERATION (2)	An operational attribute used to hold server (DSA) information that is shared among servers holding replicas of the entry
DSA_OPERATION (3)	An operational attribute used to hold server (DSA) information that is local to a server

[2.3](#) **public class LDAPAttributeSet** **implements Cloneable, Serializable, Set**

An LDAPAttributeSet is a collection of LDAPAttributes, as returned in an entry on a search or read operation, or is used to construct an entry to be added to a directory. If add() or addAll() is called and one or more of the objects to be added is not an LDAPAttribute, ClassCastException is thrown (as discussed in the documentation for java.util.Collection). To remove an attribute, remove() is called with the LDAPAttribute object to remove.

Expires December 6, 2004

[Page 25]

[2.3.1](#) Constructors

```
public LDAPAttributeSet()
```

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

[2.3.2](#) clone

```
public Object clone()
```

Returns a deep copy of this attribute set.

[2.3.3](#) getAttribute

```
public LDAPAttribute getAttribute(String attrDesc)
```

Returns the attribute matching the specified attribute description. The returned attribute has just the options specified for the given attribute type or null if none. Note: no order is implied with attribute options. Parameters are:

attrDesc	Description of the attribute. The description consists of the attribute type and any attribute options. Note: the attribute description is case insensitive and the options are not ordered. The options specify the exact set of attribute options that must be present when selecting the attribute.
----------	--

For example,

getAttribute("cn")	returns only the "cn" attribute that has no options.
getAttribute("cn;lang-en")	returns only the "cn;lang-en" attribute.
getAttribute("cn;lang-en;lang-en-us")	returns the "cn" attribute with the "lang-en-us" option and the "lang-en" option. Note: the options can be in any order.
getAttribute("cn", null)	returns all the "cn" attributes, without regard to options.
getAttribute("cn", new String[] { "lang-en" })	returns any "cn" attributes that have the lang-en attribute.

[2.3.4](#) getSubset

```
public LDAPAttributeSet getSubset(String options)
```

Expires December 6, 2004

[Page 26]

Returns a new attribute set containing only the attributes that have at least the specified options. If no attributes have the specified options, an empty LDAPAttributeSet is returned.

```
Public LDAPAttributeSet getSubset(String attrType, String options)
```

Returns a new attribute set containing only the attributes that have the specified attribute type and at least the specified options. If no attributes have the specified type and options, an empty LDAPAttributeSet is returned.

For example, suppose an attribute set contains the following attributes:

```
cn
cn;lang-ja
cn;lang-ja;phoentic      sn;lang-ja;phonetic
sn;lang-us
```

Calling the getSubset method and passing lang-ja as the argument, the method returns an attribute set containing the following attributes:

```
cn;lang-ja
sn;lang-ja;phonetic
```

Calling the getSubset method and passing type cn and lang-ja as the argument returns an attribute set containing the following attributes:

```
cn;lang-ja
cn;lang-ja;phoentic
```

Parameters are:

attrType - the attribute type of the attributes to include in the attribute set. Any options specified with this parameter are ignored. If null, all attributes matching the specified options are returned in the subset.

options - Semi-colon delimited list of subtypes to include. The options can be specified in any order. If null, all attributes of the specified type are returned. For example:

```
"lang-ja"      // The lang-ja option
```

```
"binary;lang-ja" // The binary and the lang-ja  
// options
```

Expires December 6, 2004

[Page 27]

2.4 public interface LDAPAuthHandler **extends LDAPReferralHandler**

Used by the API only if automatic referral handling is enabled in LDAPConstraints. The API ignores instances of this class if referral following is disabled (the default referral following behavior).

2.4.1 LDAPAuthHandler is used by the API to obtain credentials for reauthentication (simple bind) when automatically following a referral. If set in an LDAPConstraints instance, an application's implementation of LDAPAuthHandler is called during referral processing and returns an LDAPAuthProvider. An application may specify an instance of an LDAPConstraints class to be used on a single operation (as a method parameter) or for all operations (as connection constraints).getAuthProvider

```
public LDAPAuthProvider getAuthProvider(String host, int port)
```

Returns an object which can provide credentials to simple bind for authenticating to a server at the provided host name and port number.

Parameters are:

host	Contains a host identifier representing the IP address of a host running an LDAP server. See 2.8.9 for a discussion of valid identifiers.
port	Contains the TCP port number to connect to.

2.5 public class LDAPAuthProvider

2.5.1 Represents information the API uses to authenticate the application in cases where the the application has set an LDAPAuthHandler in LDAPConstraints to facilitate automatic referral following. Constructors

```
public LDAPAuthProvider( String dn,  
                        byte[] password )
```

Constructs information that is used by the application for simple bind authentication when following referrals automatically.

Parameters are:

Expires December 6, 2004

[Page 28]

dn	Distinguished name to use in authenticating to the server.
password	The UTF-8 or binary representation of the password to use in authenticating to the server, represented as a byte array.

[2.5.2](#) **getDN**

```
public String getDN()
```

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

[2.5.3](#) **getPassword**

```
public byte[] getPassword()
```

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

[2.6](#) **public interface LDAPBindHandler** **extends LDAPReferralHandler**

Used by the API to perform bind operations during the processing of a referral in order to follow it. If set in an LDAPConstraints instance, an application's implementation of LDAPBindHandler is called during referral processing and returns an authenticated connection to the referred server. An application may set an instance of this class in an LDAPConstraints object to be used on a single LDAP operation (as a method parameter) or for all LDAP operations (through connection constraints). An application implementing LDAPBindHandler can perform any sequence of valid LDAP operations before returning to the API, as long as it returns a connection to the referred server. If LDAPAuthHandler or LDAPBindHandler are not specified, referrals and search references followed automatically use anonymous authentication.

[2.6.1](#) **bind**

```
public LDAPConnection bind(String[] ldapurl, LDAPConnection conn)  
    throws LDAPReferralException
```

This method is called by LDAPConnection when a referral or search

continuation is received, and is responsible for binding to one of the hosts in the list specified by the ldapurl parameter (which

Expires December 6, 2004

[Page 29]

corresponds exactly to the list of hosts returned in a single referral or search continuation response). An implementation may access the host, port, socket factory and other information in the original LDAPConnection object to decide on an appropriate authentication mechanism, and/or may interact with a user or external module. The object implementing LDAPBindHandler creates a new LDAPConnection object to perform its connect and bind calls. It returns the new connection when both the connect and bind operations succeed on one host from the list. The LDAPConnection object referral following code uses the new LDAPConnection object when it resends the search request, updated with the new search base and possibly search filter. An LDAPReferralException is thrown on failure.

The API implementation dereferences the new LDAPConnection when referral following has finished, but does not call disconnect. This allows the application's implementation of LDAPBindHandler to do connection pooling when managing connections for referral following.

Parameters are:

ldapurl	List of LDAP server URLs. There is no order implied by the list.
conn	An established connection to an LDAP server.

2.7 public class LDAPCompareAttrNames implements Comparator

An object of this class defines ordering when sorting search results. When using this Comparator, LDAPEntry objects are sorted by the values of the attribute name(s) passed in the constructor, in ascending or descending order. The object is typically supplied to an implementation of the collection interfaces such as java.util.TreeSet which performs the sort.

2.7.1 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.

Expires December 6, 2004

[Page 30]

```
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:

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

[2.7.2](#) compare

```
public int compare(Object o1, Object o2)
```

Compares its two arguments for order. Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second. Throws `ClassCastException` if `o1` or `o2` is not an `LDAPEntry`.

Parameters are:

<code>o1</code>	Target entry for comparison.
<code>o2</code>	Entry to be compared to.

[2.7.3](#) equals

```
public boolean equals(Object obj)
```

Expires December 6, 2004

[Page 31]

Returns a value of true only if the specified object is also a comparator and it imposes the same ordering as this comparator.

Parameters are:

obj The reference object with which to compare.

[2.7.4](#) **getLocale**

```
public Locale getLocale()
```

Returns the Locale to be used for sorting, if a Locale has been specified. If null, a basic String.compareTo() is used for collation. If non-null, a Locale-specific collation is used.

[2.7.5](#) **setLocale**

```
public void setLocale(Locale locale)
```

Sets the Locale to be used for sorting.

Parameters are:

locale The Locale to be used for sorting.

[2.8](#) **public class LDAPConnection** **implements Cloneable**

LDAPConnection is the central class that encapsulates the connection to a Directory Server through the LDAP protocol. 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. Implementations of the API MUST ensure that methods of the LDAPConnection class are thread-safe.

[2.8.1](#) **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.

The connect or bind methods are used to connect to the LDAP server.

Expires December 6, 2004

[Page 32]


```
public LDAPConnection(SocketFactory factory)
```

Constructs a new LDAPConnection object, which will use the supplied SocketFactory class to construct a socket connection during LDAPConnection.connect(). If a security manager exists and the caller does not have permission to set a factory, SecurityException is thrown.

Parameters are:

factory	An object capable of producing a Socket.
---------	--

[2.8.2](#) abandon

```
public void abandon(LDAPSearchResults results)
                throws LDAPException
```

```
public void abandon(LDAPSearchResults results, LDAPConstraints cons)
                throws LDAPException
```

```
public void abandon(int id)
                throws LDAPException
```

```
public void abandon(int id, LDAPConstraints cons)
                throws LDAPException
```

```
public void abandon(LDAPMessageQueue queue)
                throws LDAPException
```

```
public void abandon(LDAPMessageQueue queue, LDAPConstraints cons)
                throws LDAPException
```

Either notifies the server to not send additional results associated with this LDAPSearchResults object, and discards any results already received, or abandons one or all operations for an asynchronous response queue.

If the application calls this method for a particular id or LDAPSearchResults previously abandoned, the call is ignored. An API implementation MUST ignore abandon requests for an id or LDAPSearchResults which it does not recognize. The API implementation SHOULD NOT send an additional abandon request if it can determine that one has already been sent for an id or LDAPSearchResults.

Parameters are:

results An object returned from a synchronous search.

Expires December 6, 2004

[Page 33]

id	The ID of the asynchronous operation to abandon. The ID may be obtained from the response queue for the operation.
queue	Handler returned from an asynchronous request. All outstanding operations that are managed by the queue are abandoned.
cons	Constraints specific to the operation.

[2.8.3](#) add

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

public void add(LDAPEntry entry,
    LDAPConstraints cons)
    throws LDAPException

public LDAPMessageQueue add(LDAPEntry entry,
    LDAPMessageQueue queue)
    throws LDAPException

public LDAPMessageQueue add(LDAPEntry entry,
    LDAPMessageQueue queue,
    LDAPConstraints cons)
    throws LDAPException
```

Adds an entry to the directory.

If the application does not specify attribute values which are valid according to the syntax defined for the attributes, or does not include all attributes which are required for the entry, the server will return an error.

Parameters are:

entry	LDAPEntry object specifying the distinguished name and attributes of the new entry.
queue	Handler for messages returned from a server in response to this request. If it is null, a queue object is created internally.
cons	Constraints specific to the operation.

[2.8.4](#) **addUnsolicitedNotificationListener**

```
public void addUnsolicitedNotificationListener(
```

Expires December 6, 2004

[Page 34]

LDAPUnsolicitedNotificationListener listener)

Registers an object to be notified on arrival of an unsolicited message from a server.

Parameters are:

listener	An object to be notified on arrival of an unsolicited message from a server.
----------	--

[2.8.5](#) bind (simple)

```
public void bind(int version,
                 String dn,
                 byte[] passwd)
    throws LDAPException
```

```
public void bind(int version,
                 String dn,
                 byte[] passwd,
                 LDAPConstraints cons)
    throws LDAPException
```

```
public LDAPMessageQueue bind(int version,
                              String dn,
                              byte[] passwd,
                              LDAPMessageQueue queue)
    throws LDAPException
```

```
public LDAPMessageQueue bind(int version,
                              String dn,
                              byte[] passwd,
                              LDAPMessageQueue queue,
                              LDAPConstraints cons)
    throws LDAPException
```

Synchronously authenticates using simple authentication to the LDAP server (that the object is currently connected to) using the specified name and password, with the specified LDAP protocol version. This API is specifically designed for use with LDAPv3. Unless the API provides specific support (as defined in other documents) for other versions of LDAP, version 3 should be used. If the server does not support the requested protocol version, an exception is thrown. If the object had already authenticated, the old authentication is discarded. If the object has been disconnected from an LDAP server, this method attempts to reconnect and

authenticate to the server.

Parameters are:

Expires December 6, 2004

[Page 35]

version	LDAP protocol version requested: currently 3.
dn	If the dn and passwd are non-null and non-empty, the connection and all operations through it are authenticated with dn as the distinguished name and passwd as password. If dn and/or passwd are null or empty, the connection is anonymous on completion of the simple bind request.
passwd	The UTF-8 or binary representation of the password to use in authenticating to the server, represented as a byte array. If both the passwd and dn are non-null and non-empty, the connection and all operations through it are authenticated with dn as the distinguished name and passwd as password. If dn and/or passwd is null or empty, the connection is anonymous on completion of the simple bind request.
queue	Handler for asynchronous messages returned from a server in response to this request. Ifnull, a queue object is created internally.
cons	Constraints specific to the operation.

[2.8.6](#) bind (SASL)

```
public void bind(String dn,
                 String authzId,
                 Map props,
                 javax.security.auth.callback.CallbackHandler cbh)
    throws LDAPException
```

```
public void bind(String dn,
                 String authzId,
                 Map props,
                 javax.security.auth.callback.CallbackHandler cbh,
                 LDAPConstraints cons)
    throws LDAPException
```

```
public void bind(String dn,
                 String authzId,
                 String[] mechanisms,
                 Map props,
                 javax.security.auth.callback.CallbackHandler cbh)
    throws LDAPException
```

```
public void bind(String dn,  
                  String authzId,
```

Expires December 6, 2004

[Page 36]


```
String[] mechanisms,  
Map props,  
javax.security.auth.callback.CallbackHandler cbh,  
LDAPConstraints cons)  
throws LDAPException
```

Synchronously authenticates using SASL authentication to the LDAP server (that the object is currently connected to) using the specified name and one of a specified set of mechanisms. If none of the requested SASL [[SASL](#)][AUTH][[JAVASASL](#)] mechanisms is available, an exception is thrown. If the object had already authenticated, the old authentication is discarded. If the object has been disconnected from an LDAP server, this method attempts to reconnect to the server. A SASL bind call may involve multiple protocol requests and responses. An attempt to invoke an operation other than bind or unbind between bind requests in a multi-stage bind, results in an LDAPException with the result code SASL_BIND_IN_PROGRESS. Parameters are:

dn	The distinguished name to use as the bind name. It may be null or empty. This value is not used as either a SASL authentication nor authorization identity. The application provides these identities through the callback handler.
authzId	If not null and not empty, an LDAP authzID [AUTH] to be passed to the SASL layer. If null or empty, the authzId will be treated as an empty string and processed as per RFC 2222 [SASL].
mechanisms	An array of IANA-registered SASL mechanisms which the client application is willing to use for authentication. Null or an empty array may be specified to abort the negotiation, forcing the server to return an AUTH_METHOD_NOT_SUPPORTED result.
props	Optional qualifiers for the authentication session.
cbh	A class which may be called by the SASL client implementation to obtain additional information required, such as additional credentials.
cons	Constraints specific to the operation.

See [[JAVASASL](#)] for additional information about the above parameters.

Expires December 6, 2004

[Page 37]

2.8.7 clone

```
public Object 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.

The clone can freely modify options and search constraints, and issue requests, without affecting the source object or other clones. If the clone disconnects or reconnects, it is completely dissociated from the source object and other clones. Reauthenticating in a clone, however, is a global operation which will affect the source object and all associated clones, because it applies to the single shared physical connection. Any request by an associated object after one has reauthenticated will carry the new identity.

Methods that are global in nature and which affect the source object are:

```
addUnsolicitedNotificationListener  
bind  
connect  
disconnect  
finalize  
removeUnsolicitedNotificationListener  
startTLS
```

The following methods return data that is from the source object and is the same for all clones of LDAPConnection:

```
getAuthenticationDN  
getAuthenticationMethod  
getHost  
getPort  
getProtocolVersion  
getSaslBindCallbackHandler  
getSaslBindProperties  
getSocketFactory  
isBound  
isConnected  
isTLS
```

The following methods manipulate or retrieve data that is unique to each clone of LDAPConnection:

```
getConstraints
```

getResponseControls
getSearchConstraints
setConstraints

Expires December 6, 2004

[Page 38]

[2.8.8](#) compare

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

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

public LDAPMessageQueue compare(String dn,
                               LDAPAttribute attr,
                               LDAPMessageQueue queue)
                               throws LDAPException

public LDAPMessageQueue compare(String dn,
                               LDAPAttribute attr,
                               LDAPMessageQueue queue,
                               LDAPConstraints cons)
                               throws LDAPException
```

Checks to see if an entry in the Directory Server contains an attribute with a specified value. The synchronous methods return a value of true if the entry has the value, and false if the entry does not have the value or the attribute. The method throws an `IllegalArgumentException` if `LDAPAttribute` object specified by the `attr` parameter contains more than one value.

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.
queue	Handler for messages returned from a server in response to this request. If it is null, a queue object is created internally.
cons	Constraints specific to the operation.

[2.8.9](#) connect

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

Expires December 6, 2004

[Page 39]

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.

Parameters are:

host	Contains a host identifier consisting of a hostname, an IPv4 dotted string, or an IPv6 reference [IPv6] representing the IP address of a host running an LDAP server to connect to. Alternatively, it may contain a list of host identifiers, space-delimited. Each host identifier may include a trailing colon and port number. IPv6 identifiers with a port number are represented with square brackets around the IP address part as per [IPv6URL]. In the case where more than one host identifier is specified, each host identifier in turn will be contacted until a connection can be established. Examples:
------	---

"directory.example.com"

"192.0.2.0"

"[FEDC:BA98:7654:3210:FEDC:BA98:7654:3210]:4389"

"directory.example.com:1050 people.catalog.com 192.0.2.0"

port	Port number for LDAP server (use LDAPConnection.DEFAULT_PORT for default port). "port" is ignored for any host identifier which includes a colon and port number.
------	---

[2.8.10](#) delete

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

```
public void delete(String dn,  
                  LDAPConstraints cons)  
    throws LDAPException
```

```
public LDAPMessageQueue delete(String dn,  
                               LDAPMessageQueue queue)  
    throws LDAPException
```

```
public LDAPMessageQueue delete(String dn,  
                               LDAPMessageQueue queue,
```

LDAPConstraints cons)
throws LDAPException

Expires December 6, 2004

[Page 40]

Deletes the entry for the specified DN from the directory.

Parameters are:

dn	Distinguished name of the entry to delete.
queue	Handler for messages returned from a server in response to this request. If it is null, a queue object is created internally.
cons	Constraints specific to the operation.

[2.8.11](#) **disconnect**

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

```
public void disconnect(LDAPConstraints cons) throws LDAPException
```

Disassociates the LDAPConnection object from clones and any physical connection to an LDAP server. If the object is the last clone sharing a physical connection, the method closes the connection with the LDAP server. The API implementation sends an Unbind request to the server with any controls specified by the LDAPConstraints object before closing the connection. Before the application can perform LDAP operations again, it **MUST** reconnect to a server by calling either connect or bind (bind will attempt to reconnect to the previous server).

Parameters are:

cons	Constraints to be sent with the unbind request.
------	---

[2.8.12](#) **extendedOperation**

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

```
public LDAPExtendedResponse extendedOperation(  
    LDAPExtendedOperation op,  
    LDAPConstraints cons )  
    throws LDAPException
```

```
public LDAPMessageQueue extendedOperation(  
    LDAPExtendedOperation op,
```

LDAPMessageQueue queue)
throws LDAPException

Expires December 6, 2004

[Page 41]

```
public LDAPMessageQueue extendedOperation(  
    LDAPExtendedOperation op,  
    LDAPConstraints cons,  
    LDAPMessageQueue queue)  
    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 OID and an Octet String or BER-encoded value(s).

Parameters are:

op	Object which contains the OID of the extended operation and any operation-specific data.
cons	Constraints specific to the operation.

[2.8.13](#) **fetchSchema**

```
public LDAPSchema fetchSchema(String schemaDN)  
    throws LDAPException
```

Retrieves the schema associated with a particular schema DN in the Directory Server. The schema DN for a particular entry is obtained by calling the getSchemaDN method of LDAPConnection (see 2.8.25).

An LDAPException is thrown if the schema cannot be retrieved.

Parameters are:

schemaDN	The schema DN used to fetch the schema.
----------	---

[2.8.14](#) **finalize**

```
protected void finalize() throws LDAPException
```

Closes the connection if open and releases any other resources held by the object.

[2.8.15](#) **getAuthenticationDN**

```
public String getAuthenticationDN()
```

Returns the distinguished name (DN) used as the bind name during the last successful bind operation. null is returned if no authentication

has been performed or if the bind resulted in an anonymous connection.

Expires December 6, 2004

[Page 42]

2.8.16 getAuthenticationMethod

```
public String getAuthenticationMethod()
```

Returns the method used to authenticate the connection. The return value is one of the following:

"none"	The current authentication state has not been established by use of the bind operation. This is the initial state upon connect(), as well as if the last bind failed.
"simple"	Simple bind has completed successfully (anonymous, unauthenticated, or authenticated)
"sasl"	The current authentication state was established by the successful completion of a SASL bind

2.8.17 getConstraints

```
public LDAPConstraints getConstraints()
```

Returns a copy of the set of constraints associated with this connection. These constraints apply to all operations performed through this connection (unless a different set of constraints is specified when calling an operation method). If no constraints have been assigned with setConstraints, a copy of the default constraints is returned.

2.8.18 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. If no connection attempt has been made, null is returned.

2.8.19 getPort

```
public int getPort()
```

Returns the port number of the LDAP server to which the object is or was last connected. If no connection attempt has been made, LDAPConnection.DEFAULT_PORT is returned.

Expires December 6, 2004

[Page 43]

2.8.20 getProperty

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

Gets a property of a connection object. The properties are defined by the API implementation and not modifiable by the client application.

Parameters are:

name	Name of the property to be returned.
------	--------------------------------------

The following read-only properties are available for any given connection:

LDAP_PROPERTY_SDK ("version.sdk")	The version of this SDK, as a String data type.
-----------------------------------	---

LDAP_PROPERTY_PROTOCOL ("version.protocol")	The highest supported version of the LDAP protocol, as an Integer data type.
---	--

LDAP_PROPERTY_SECURITY ("security.types")	A comma-separated list of the types of authentication supported, as a String. See 2.8.16.
---	---

Other properties MAY be available in particular implementations of the class.

A deep copy of the property is provided where applicable; the client application does not need to clone the object received.

null is returned if the requested property is not available.

2.8.21 getProtocolVersion

```
public int getProtocolVersion ()
```

Returns the protocol version that the connection is bound to (which currently is 3). If the connection is not bound, it returns 3.

Expires December 6, 2004

[Page 44]

[2.8.22](#) **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. For asynchronous requests, the response controls are available in `LDAPMessage` instead. Returns null if none or if using asynchronous requests.

[2.8.23](#) **getSaslBindCallbackHandler**

```
public javax.security.auth.callback.CallbackHandler  
    getSaslBindCallbackHandler()
```

Returns the callback handler, if any, specified on binding with a SASL mechanism, or null if none.

[2.8.24](#) **getSaslBindProperties**

```
public Map getSaslBindProperties()
```

Returns the properties, if any, specified on binding with a SASL mechanism, or null if none.

[2.8.25](#) **getSchemaDN**

```
public String getSchemaDN() throws LDAPException
```

Retrieves the DN for the schema at the root DSE of the Directory Server.

Throws `LDAPException` if the schema DN cannot be retrieved, or if the `subschemaSubentry` attribute associated with the root DSE contains multiple values.

```
public String getSchemaDN(String dn) throws LDAPException
```

Retrieves the DN of the schema associated with a particular entry in the directory. Used with `LDAPConnection.fetchSchema()`, see 2.8.13.

Throws `LDAPException` if the schema DN cannot be retrieved, or if a null or empty value is passed as `dn`, or if the `subschemaSubentry` attribute associated with the root DSE contains multiple values.

Parameters are:

Expires December 6, 2004

[Page 45]

dn Distinguished name of the entry for which the schema DN is to be retrieved.

[2.8.26](#) **getSearchConstraints**

```
public LDAPSearchConstraints getSearchConstraints()
```

Returns a clone of the search constraints associated with this connection. These constraints apply to search operations performed through this connection (unless a different set of constraints is specified when calling the search operation method). The search constraints include the base constraints returned by `getConstraints()`. If no constraints have been assigned with `setConstraints`, a clone of the default constraints is returned.

[2.8.27](#) **getSocketFactory**

```
public SocketFactory getSocketFactory()
```

Returns the `SocketFactory` used to establish a connection to a server.

[2.8.28](#) **isBound**

```
public boolean isBound()
```

Indicates whether the object has authenticated to the connected LDAP server (other than anonymously with simple bind). It returns `false` initially, `false` upon a bind request, and `true` after successful completion of the last outstanding non-anonymous simple bind.

[2.8.29](#) **isConnected**

```
public boolean isConnected()
```

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

[2.8.30](#) **isTLS**

```
public boolean isTLS ()
```

Indicates the session is currently protected by TLS. The method provides no indication of the level of protection provided.

[2.8.31](#) modify

Expires December 6, 2004

[Page 46]

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

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

public LDAPMessageQueue modify(String dn,
                              LDAPModification mod,
                              LDAPMessageQueue queue)
    throws LDAPException

public LDAPMessageQueue modify(String dn,
                              LDAPModification mod,
                              LDAPMessageQueue queue,
                              LDAPConstraints cons)
    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.

The application is responsible for specifying attribute values which are valid according to the syntax defined for the attributes.

```
public void modify(String dn,
                  LDAPModification[] mods)
    throws LDAPException

public void modify(String dn,
                  LDAPModification[] mods,
                  LDAPConstraints cons)
    throws LDAPException

public LDAPMessageQueue modify(String dn,
                              LDAPModification[] mods,
                              LDAPMessageQueue queue)
    throws LDAPException

public LDAPMessageQueue modify(String dn,
                              LDAPModification[] mods,
                              LDAPMessageQueue queue,
                              LDAPConstraints cons)
```

throws LDAPException

Makes multiple changes to an existing entry in the directory (for

Expires December 6, 2004

[Page 47]

example, changes attribute values, adds new attribute values, or removes existing attribute values).

The application is responsible for specifying attribute values which are valid according to the syntax defined for the attributes.

Parameters are:

dn	Distinguished name of the entry to modify.
mod	A single change to be made to the entry.
mods	An array specifying multiple changes to be made to the entry. The changes are made in the order specified.
queue	Handler for messages returned from a server in response to this request. If it is null, a queue object is created internally.
cons	Constraints specific to the operation.

[2.8.32](#) read

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

```
public LDAPEntry read(String dn,  
                      LDAPSearchConstraints cons)  
                      throws LDAPException
```

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

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

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

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

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

Expires December 6, 2004

[Page 48]


```
public static LDAPEntry read(LDAPUrl toGet,  
                             LDAPSearchConstraints cons)  
    throws LDAPException
```

Reads the entry specified by the LDAP URL and retrieves all attributes for the entry.

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

If the URL specifies a scope other than base, `IllegalArgumentException` is thrown. Any critical extensions specified in the URL must be processed or else an `LDAPException` is thrown with the result code `UNSUPPORTED_OPERATION`.

The method returns the entry specified by the base DN.

Parameters are:

dn	Distinguished name of the entry to retrieve.
cons	Constraints specific to the operation.
attrs	Names of attributes to retrieve.
toGet	LDAP URL specifying the entry to read.

If the server does not return exactly one entry, an `LDAPException` is thrown with a result code of `AMBIGIOUS_RESPONSE`.

Note: `read` is simply a helper method and uses the `ldap search` operation to achieve the results. As such, there is no asynchronous interface.

[2.8.33](#) `removeUnsolicitedNotificationListener`

```
public void removeUnsolicitedNotificationListener(  
    LDAPUnsolicitedNotificationListener listener)
```

Deregisters an object so that it will no longer be notified on arrival of an unsolicited message from a server. If the object is null or was not previously registered for unsolicited notifications, the method does nothing.

Parameters are:

Expires December 6, 2004

[Page 49]

listener An object to no longer be notified on arrival of
 an unsolicited message from a server.

[2.8.34](#) rename

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

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

public LDAPMessageQueue rename(String dn,
                               String newRdn,
                               boolean deleteOldRdn,
                               LDAPMessageQueue queue)
    throws LDAPException

public LDAPMessageQueue rename(String dn,
                               String newRdn,
                               boolean deleteOldRdn,
                               LDAPMessageQueue queue,
                               LDAPConstraints cons)
    throws LDAPException
```

Renames an existing entry in the directory.

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

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

public LDAPMessageQueue rename(String dn,
                               String newRdn,
                               String newParentdn,
```

```
boolean deleteOldRdn,  
LDAPMessageQueue queue)  
throws LDAPException
```

Expires December 6, 2004

[Page 50]

```
public LDAPMessageQueue rename(String dn,
                               String newRdn,
                               String newParentdn,
                               boolean deleteOldRdn,
                               LDAPMessageQueue queue,
                               LDAPConstraints cons)
    throws LDAPException
```

Renames an existing entry or subtree 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. If newParentdn is null, the request is treated as if the method without newParentdn is called.
deleteOldRdn	If true, the old name is not retained as an attribute value.
queue	Handler for messages returned from a server in response to this request. If it is null, a queue object is created internally.
cons	Constraints specific to the operation.

[2.8.35](#) search

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

```
public LDAPMessageQueue search(String base,
                                int scope,
                                String filter,
                                String[] attrs,
                                boolean typesOnly,
                                LDAPMessageQueue queue)
    throws LDAPException
```

Performs the search specified by the parameters.

Expires December 6, 2004

[Page 51]

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

```
public LDAPMessageQueue search(String base,
                                int scope,
                                String filter,
                                String[] attrs,
                                boolean typesOnly,
                                LDAPMessageQueue queue,
                                LDAPSearchConstraints cons)
    throws LDAPException
```

Performs the search specified by the parameters, also allowing specification of operation specific 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 operation or default search constraints, a choice can be made as to 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 received from the server.

```
public static LDAPSearchResults search(LDAPUrl toGet)
    throws LDAPException
```

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

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

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

When the methods using the LDAPUrl parameter are called, a new

connection is created automatically, using the host and port specified in the URL. After all search results have been received

Expires December 6, 2004

[Page 52]

from the server, the method closes the connection (in other words, it disconnects from the LDAP server).

As part of operation or default 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 received from the server.

Parameters are:

base	The base distinguished name to search from.
scope	The scope of the entries to search. The following are the valid options: SCOPE_BASE Search only the base DN SCOPE_ONE Search only entries directly under the base DN SCOPE_SUB Search the base DN and all entries within its subtree
filter	Search filter specifying the search criteria, as defined in [FILTER]. The value null can be passed to indicate that the filter "(objectclass=*)" which matches all entries is to be used.
attrs	Names of attributes to retrieve. If null or an empty array is specified, all attributes are retrieved.
typesOnly	If true, returns the names but not the values of the attributes found. If false, returns the names and values for attributes found.
toGet	LDAP URL specifying the entry to read.
queue	Handler for messages returned from a server in response to this request. If it is null, a queue object is created internally.
cons	Constraints specific to the search.

Note: [RFC 2251](#) [[LDAPPROTO](#)] indicates that extendedResponses on search requests may be defined in future versions of the LDAP protocol. There is no support for extendedResponses on search requests in this

version of the Java LDAP API.

Expires December 6, 2004

[Page 53]

2.8.36 setConstraints

```
public void setConstraints(LDAPConstraints cons)
```

Sets the constraints that apply to all operations performed through this connection (unless a different set of constraints is specified when calling an operation method). An LDAPSearchConstraints object which is passed to this method will override all constraints (search and base), while an LDAPConstraints object will only affect the base constraints.

Parameters are:

cons	Non-null constraints object.
------	------------------------------

2.8.37 setSocketFactory

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

Establishes the default SocketFactory used when LDAPConnection objects are constructed unless an SocketFactory is specified in the LDAPConnection object constructor.

This method sets the default SocketFactory used for all subsequent LDAPConnection objects constructed. If called after LDAPConnection objects are created, those already created are not affected even if they disconnect and establish a new connection. It affects LDAPConnection objects only as they are constructed.

If a security manager exists and the caller does not have permission to set a factory, SecurityException is thrown.

To use the setSocketFactory method, the caller needs the following permission:

```
java.lang.RuntimePermission("setFactory");
```

Parameters are:

factory	A factory object which can construct socket connections for an LDAPConnection. If null, the default factory of the API implementation is selected.
---------	--

2.8.38 startTLS

```
public void startTLS()
```

throws LDAPException

Expires December 6, 2004

[Page 54]

Begin using the Transport Layer Security (TLS) protocol for session privacy [[TLS](#)][LDAPTLS]. If the socket factory of the connection is not capable of initiating a TLS session, an `LDAPException` is thrown with the error code `TLS_NOT_SUPPORTED`. If the server does not support the transition to a TLS session, an `LDAPException` is thrown with the error code returned by the server. If there are outstanding LDAP operations on the connection, an `LDAPException` is thrown.

[2.8.39](#) stopTLS

```
public void stopTLS ()  
throws LDAPException
```

Stop using the Transport Layer Security (TLS) protocol for session privacy [[LDAPTLS](#)]. If the server does not support the termination of a TLS session, an `LDAPException` is thrown with the error code returned by the server. If there are outstanding LDAP operations on the connection, an `LDAPException` is thrown.

[2.8.40](#) Constants of LDAPConnection

`ALL_USER_ATTRS ("*")` Used with search in an attribute list to indicate that all attributes (other than operational attributes) are to be returned.

`NO_ATTRS ("1.1")` Used with search instead of an attribute list to indicate that no attributes are to be returned.

`DEFAULT_PORT (389)` Used with connect to indicate the default LDAP port number.

`SCOPE_BASE (0)` Used with search to indicate that only the entry corresponding to the base DN is to be returned.

`SCOPE_ONE (1)` Used with search to indicate that only immediate subordinates of the entry corresponding to the base DN, and not the entry corresponding to the base DN, are to be returned.

`SCOPE_SUB (2)` Used with search to indicate that the entry corresponding to the base DN as well as all direct and indirect subordinate entries are to be returned.

`LDAP_PROPERTY_SDK ("version.sdk")` Used with

getProperty to retrieve the version of the SDK.

Expires December 6, 2004

[Page 55]

LDAP_PROPERTY_PROTOCOL ("version.protocol") Used with
getProperty to retrieve the highest supported
LDAP protocol version.

LDAP_PROPERTY_SECURITY ("security.types") Used with
getProperty to retrieve a list of the
authentication types supported.

2.9 public class LDAPConstraints **implements Cloneable, Serializable**

A set of options to control any operation. There is always an LDAPConstraints associated with an LDAPConnection object; its values can be changed with LDAPConnection.setConstraints, or overridden by passing an LDAPConstraints object to an operation.

2.9.1 Constructors

```
public LDAPConstraints()
```

Constructs an LDAPConstraints object that specifies the default set of constraints.

```
public LDAPConstraints(int msLimit,  
                      boolean doReferrals,  
                      LDAPReferralHandler handler,  
                      int hop_limit)
```

Constructs a new LDAPConstraints object and allows specifying the operational 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). This is an interface-enforced limit.
doReferrals	Specify true to follow referrals automatically, or false to throw an LDAPReferralException error if the server sends back a referral (false by default). It is ignored for asynchronous operations.
handler	Custom authentication processor, called when the LDAPConnection needs to authenticate, typically

on following a referral. The value null indicates default authentication processing, which is, to use anonymous authentication if automatically

Expires December 6, 2004

[Page 56]

following referrals. This parameter ignored if `doReferrals` is set to false. The handler object may implement either the `LDAPBindHandler` or the `LDAPAuthHandler` interface. Any settings for following referrals are ignored for asynchronous operations.

<code>hop_limit</code>	Maximum number of referrals to follow in a sequence when attempting to resolve a request, when doing automatic referral following. It is ignored for asynchronous operations.
------------------------	---

[2.9.2](#) **getControls**

```
public LDAPControl[] getControls()
```

Returns controls to be sent to the server.

[2.9.3](#) **getHopLimit**

```
public int getHopLimit()
```

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

[2.9.4](#) **getProperty**

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

Gets a property of a constraints object which has been assigned with `setProperty`. null is returned if the property is not defined.

Parameters are:

<code>name</code>	Name of the property to be returned.
-------------------	--------------------------------------

[2.9.5](#) **getReferralFollowing**

```
public boolean getReferralFollowing()
```

Specifies whether or not referrals are followed automatically. Returns true if referrals are to be followed automatically, or false if receipt of referrals causes the API to throw an `LDAPReferralException`.

[2.9.6](#) `getTimeLimit`

Expires December 6, 2004

[Page 57]

```
public int getTimeLimit()
```

Returns the maximum number of milliseconds the client application waits for any operation under these constraints. If 0, there is no maximum time limit on waiting for the operation results. The time limit is enforced by the API, and the actual granularity of the timeout depends on the implementation.

[2.9.7](#) **setControls**

```
public void setControls( LDAPControl control )
```

```
public void setControls( LDAPControl[] controls )
```

Sets controls to be sent to the server. If controls are not set by calling this method, no controls are sent to the server.

Parameters are:

control A single control to be sent to the server.

controls An array of controls to be sent to the server.

[2.9.8](#) **setHopLimit**

```
public void setHopLimit(int hop_limit)
```

Sets the maximum number of hops to follow in sequence during automatic referral following. The default is 10. 0 means no limit.

Parameters are:

hop_limit Maximum number of chained referrals to follow automatically.

[2.9.9](#) **setProperty**

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

Sets a property of the constraints 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 and proprietary extensions to operation modifiers. Throws `IllegalArgumentException` if the property name is not defined in the API.

Parameters are:

Expires December 6, 2004

[Page 58]

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

[2.9.10](#) **setReferralFollowing**

```
public void setReferralFollowing(boolean doReferrals)
```

Specifies whether or not referrals are followed automatically, or if referrals throw an `LDAPReferralException`. The default is false, i.e., do not follow referrals

Parameters are:

doReferrals	True to follow referrals automatically.
-------------	---

[2.9.11](#) **setReferralHandler**

```
public void setReferralHandler (LDAPReferralHandler handler)
```

Specifies the object that will process authentication requests. The default is null.

Parameters are:

handler	An object that implements <code>LDAPBindHandler</code> or <code>LDAPAuthHandler</code> .
---------	--

[2.9.12](#) **setTimeLimit**

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

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

Parameters are:

msLimit	Maximum milliseconds to wait.
---------	-------------------------------

[2.10](#) **public class LDAPControl** **implements Serializable, Cloneable**

An `LDAPControl` encapsulates optional additional parameters or

constraints to be applied to LDAP operations. When included with
LDAPConstraints or LDAPSearchConstraints on an LDAPConnection or on a

Expires December 6, 2004

[Page 59]

specific operation request, it is sent to the server along with operation requests.

[2.10.1](#) Constructors

```
public LDAPControl(String oid,  
                   boolean critical,  
                   byte[] value)
```

Parameters are:

oid	The type of the Control, as an OID.
critical	If true, the LDAP operation will fail with UNAVAILABLE_CRITICAL_EXTENSION if the server does not support this Control.
value	BER-Encoded control-specific data. The API implementation does not interpret or convert the value.

[2.10.2](#) clone

```
public Object clone()
```

Returns a deep copy of the object.

[2.10.3](#) getID

```
public String getID()
```

Returns the OID of the control.

[2.10.4](#) getValue

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

Returns the control-specific data of the object.

[2.10.5](#) isCritical

```
public boolean isCritical()
```

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

Expires December 6, 2004

[Page 60]

2.10.6 register

```
public static void register(String oid, Class controlClass)
```

Registers an application class to be instantiated on receipt of a control with the given OID. Any previous registration for the OID is overridden. The controlClass MUST be an extension of LDAPControl.

Parameters are:

oid	The OID of the Control.
controlClass	A class which can instantiate an LDAPControl.

2.10.7 setValue

```
protected void setValue(byte[] value)
```

Sets the BER-Encoded control-specific data of the object. This method is for use by extensions of LDAPControl.

Parameters are:

value	The value to be assigned to the Control.
-------	--

2.11 public class LDAPDITContentRuleSchema extends LDAPSchemaElement

The LDAPDITContentRuleSchema class represents the definition of a DIT Content Rule. It is used to discover or modify additional auxiliary classes, mandatory and optional attributes, and restricted attributes in effect for an object class. See [\[ATTR\]](#) for a description of DIT content rule representation in LDAP.

2.11.1 Constructors

```
public LDAPDITContentRuleSchema(String[] names,  
                                String oid,  
                                String description,  
                                boolean obsolete,  
                                String[] auxiliary,  
                                String[] required,  
                                String[] optional,  
                                String[] precluded)
```

Constructs a DIT content rule for adding to or deleting from the schema. [\[LDAPPROTO\]](#) defines which parameters are optional (may be

null).

Expires December 6, 2004

[Page 61]

```
public LDAPDITContentRuleSchema(String raw)
```

Constructs a DIT content rule from an encoding using the ditContentRules syntax [[ATTR](#)].

Parameters are:

names	Name(s) of the content rule.
oid	OID of the content.
description	Optional description of the content rule.
obsolete	true if the content rule is obsolete.
auxiliary	A list of auxiliary object classes allowed for an entry to which this content rule applies. These may either be specified by name or by OID.
required	A list of user attribute types that an entry to which this content rule applies must contain in addition to its normal set of mandatory attributes. These may either be specified by name or OID.
optional	A list of user attribute types that an entry to which this content rule applies may contain in addition to its normal set of optional attributes. These may either be specified by name or OID.
precluded	A list, consisting of a subset of the optional user attribute types of the structural and auxiliary object classes which are precluded from an entry to which this content rule applies. These may either be specified by name or OID.
raw	A DIT content rule encoded using the ditContentRules syntax [ATTR].

[2.11.2](#) getAuxiliaryClasses

```
public String[] getAuxiliaryClasses()
```

Returns the list of allowed auxiliary classes.

[2.11.3](#) `getOptionalAttributes`

```
public String[] getOptionalAttributes()
```

Expires December 6, 2004

[Page 62]

Returns the list of additional optional attributes for an entry controlled by this content rule.

[2.11.4](#) **getPrecludedAttributes**

```
public String[] getPrecludedAttributes()
```

Returns the list of precluded attributes for an entry controlled by this content rule.

[2.11.5](#) **getRequiredAttributes**

```
public String[] getRequiredAttributes()
```

Returns the list of additional required attributes for an entry controlled by this content rule.

[2.12](#) **public class LDAPDITStructureRuleSchema** **extends LDAPSchemaElement**

The LDAPDITStructureRuleSchema class represents the definition of a DIT Structure Rule. It is used to discover or modify which object classes a particular object class may be subordinate to in the DIT. See [\[ATTR\]](#) for a description of DIT structure rule representation in LDAP.

[2.12.1](#) **Constructors**

```
public LDAPDITStructureRuleSchema(String[] names,  
                                   int ruleID,  
                                   String description,  
                                   boolean obsolete,  
                                   String nameForm,  
                                   String[] superiorIDs)
```

Constructs a DIT structure rule for adding to or deleting from the schema. [\[LDAPPROTO\]](#) defines which parameters are optional (may be null).

```
public LDAPDITStructureRuleSchema(String raw)
```

Constructs a DIT structure rule from an encoding using the `dITStructureRules` syntax [\[ATTR\]](#).

Parameters are:

names

Name(s) of the structure rule.

Expires December 6, 2004

[Page 63]

ruleID	Unique identifier of the structure rule. NOTE: this is an integer, not an OID. Structure rules aren't identified by OID.
description	Optional description of the structure rule.
obsolete	true if the structure rule is obsolete.
nameForm	Either the OID or the name of a name form. This is used to indirectly refer to the object class that this structure rule applies to.
superiorIDs	List of superior structure rules - specified by their integer ID, or null if none. The object class specified by this structure rule (via the nameForm parameter) may only be subordinate in the DIT to object classes of those represented by the structure rules here .
raw	A DIT structure rule encoded using the dITStructureRules syntax [ATTR].

[2.12.2](#) getNameForm

```
public String getNameForm()
```

Returns the NameForm that this structure rule controls. You can get the actual object class that this structure rule controls by calling `getNameForm(ditStructRule.getNameForm()).getObjectClass()`.

[2.12.3](#) getRuleID

```
public int getRuleID()
```

Returns the rule ID for this structure rule. Note that this returns an integer rather than an OID. Objects of this class do not have an OID, thus `getID` will return null.

[2.12.4](#) getSuperiors

```
public String[] getSuperiors()
```

Returns a list of all structure rules that are superior to this structure rule. To resolve to an object class, you need to first resolve the superior id to another structure rule, then call

`getNameForm().getObjectContext()` on that structure rule.

Expires December 6, 2004

[Page 64]

2.13 public class LDAPDN

A utility class used to manipulate a distinguished name (DN).

2.13.1 equals

```
public static boolean equals(String dn1, String dn2)
```

Compares the two strings per the distinguishedNameMatch matching rule [\[ATTR\]](#). An API implementation MUST use caseIgnoreMatch equality matching for the attributes listed in section 2 of [\[ATTR\]](#). `IllegalArgumentException` is thrown if one or both DNs are invalid. `UnsupportedOperationException` is thrown if the API implementation is not able to determine if the DNs match or not.

Parameters are:

dn1	String form of first DN to compare.
dn2	String form of second DN to compare.

2.13.2 escapeRDN

```
public static String escapeRDN(String rdn)
```

Returns the RDN after escaping the characters requiring escaping [\[DN\]](#). For example, for the rdn "cn=Example, Inc", "cn=Example\, Inc" is returned.

Parameters are:

rdn	The RDN to escape.
-----	--------------------

2.13.3 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,dc=example,dc=com"
noTypes	If true, returns only the values of the components, and not the names, e.g. "Babs"

Jensen", "Accounting", "Example", "com" - instead

Expires December 6, 2004

[Page 65]

of "cn=Babs Jensen","ou=Accounting","dc=Example",
and "dc=com".

[2.13.4](#) 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 component of a distinguished name.
noTypes	If true, returns only the values of the components, and not the names.

[2.13.5](#) isValid

```
public static boolean isValid(String dn)
```

Returns true if the string conforms to distinguished name syntax (section 3 of [\[DN\]](#), or if the string conforms to [section 4](#) of [\[DN\]](#).

Parameters are:

dn	String to evaluate for distinguished name syntax.
----	---

[2.13.6](#) normalize

```
public static String normalize(String dn)
```

Returns the DN normalized by removal of non-significant space characters as per [RFC 2253, section 4](#) [\[DN\]](#).

Parameters are:

dn	The DN to normalize.
----	----------------------

[2.13.7](#) unescapeRDN

```
public static String unescapeRDN(String rdn)
```

Returns the RDN after unescaping the characters requiring escaping [\[DN\]](#). For example, for the rdn "cn=Example\, Inc", "cn=Example, Inc"

is returned. `IllegalArgumentException` is thrown if the RDN cannot be parsed.

Expires December 6, 2004

[Page 66]

Parameters are:

rdn	The RDN to unescape.
-----	----------------------

2.14 public class LDAPEntry **implements Serializable, Comparable**

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 either enumerating an LDAPSearchResults, or calling LDAPSearchResult.getEntry.

2.14.1 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.

2.14.2 compareTo

```
public int compareTo(Object obj)
```

Compares this object with the specified object for order. Ordering is

determined by comparing normalized DN values (see LDAPEntry.getDN()
2.14.5 and LDAPDN.normalize() 2.13.6) using the compareTo() method of

Expires December 6, 2004

[Page 67]

the `String` class. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters are:

<code>obj</code>	The object to be compared to this object.
------------------	---

[2.14.3](#) `getAttribute`

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

Returns a copy of the attribute matching the specified `attrName` or null if none.

Parameters are:

<code>attrName</code>	The name of the attribute. See 2.3.3 for the syntax and semantics relevant to this parameter.
-----------------------	---

[2.14.4](#) `getAttributeSet`

```
public LDAPAttributeSet getAttributeSet()
```

Returns a copy of the attribute set of the entry. Copies of all base and option variants of all attributes are returned. The `LDAPAttributeSet` returned is empty if there are no attributes in the entry.

```
public LDAPAttributeSet getAttributeSet(String options)
```

Returns an attribute set from the entry, consisting of copies of only those attributes matching the specified `options(s)`. "option" may be, for example, "lang-ja", "binary", or "lang-ja;phonetic". If more than one option is specified, separated with a semicolon, only those attributes with all of the named options will be returned. The `LDAPAttributeSet` returned may be empty if there are no matching attributes in the entry.

Parameters are:

<code>option</code>	One or more option specification(s), separated with semicolons. "lang-ja" and "lang-en;phonetic" are valid option specifications.
---------------------	---

[2.14.5](#) getDN

Expires December 6, 2004

[Page 68]


```
public String getDN()
```

Returns the distinguished name of the entry.

2.15 public class LDAPException extends Exception

Thrown to indicate an error has occurred. An LDAPException typically results from errors reported by the directory server. Errors not reported by the directory server (such as network errors, or invalid usage of the API) are thrown as LDAPLocalException objects (see also 2.18 and 2.26).

The
getLDAPResultCode() method returns the specific LDAP result code.

2.15.1 Constructors

```
public LDAPException()
```

Constructs a default exception with no specific error information.

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

Constructs an exception with a string describing the error, a result code, and optionally a message from the server.

```
public LDAPException(String message,  
                     int resultCode,  
                     String serverMessage,  
                     Throwable rootException)
```

Constructs an exception with a string describing the error, a result code, an optional message from the server, and an embedded root exception as additional information.

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

Constructs an exception with a string describing the error, a result code, an optional message from the server, and the maximal subset of a specified DN which could be matched by the server on a search

operation.

Expires December 6, 2004

[Page 69]

Parameters are:

message	The descriptive error string or null if none
resultCode	The result code returned
serverMessage	Error message specifying additional information from the server or null if none
matchedDN	The DN of the most immediate ancestor of a specified search DN which could be found by the server on a search operation or null if none
rootException	An exception which caused the failure or null if none

[2.15.2](#) **getCause**

```
public Throwable getCause()
```

Returns the lower level Exception which caused the failure, or null if none. For example, an IOException with additional information may be returned on a CONNECT_ERROR failure.

[2.15.3](#) **getLDAPErrorMessage**

```
public String getLDAPErrorMessage()
```

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

[2.15.4](#) **getResponse**

```
public LDAPMessage getResponse()
```

Returns the LDAPMessage of a response received as a result of an LDAP Intermediate Response message, or from a response received with an unknown LDAP Message Type. If this exception does not have the result code of INTERMEDIATE_RESPONSE or UNKNOWN_TYPE, this method returns null.

[2.15.5](#) **getMatchedDN**

```
public String getMatchedDN()
```

Expires December 6, 2004

[Page 70]

Returns the `matchedDN` value provided by the server in the response which generated this exception (it may be empty). If the exception is not due to a server response, null is returned.

2.15.6 getResultCode

```
public int getResultCode()
```

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 [[LDAPPROTO](#)]. Otherwise, if the exception was generated by the API implementation, a local error code is returned (see "Result codes" at 2.15.9) and the exception class is an instance of `LDAPLocalException`, see 2.18.

2.15.7 resultCodeToString

```
public String resultCodeToString()
```

Returns a String representing the result code in the default Locale.

```
public static String resultCodeToString( int code )
```

Returns a String representing the specified result code in the default Locale, or null if there is no such code known to the API.

```
public String resultCodeToString( Locale locale )
```

Returns a String representing the result code in the specified Locale, or null if a String representation is not available for the requested Locale.

```
public static String resultCodeToString( int code, Locale locale )
```

Returns a String representing the specified result code in the specified Locale, or null if there is no such code or if a String representation is not available for the requested Locale.

Parameters are:

code	One of the result codes listed in "Result codes" below.
------	---

locale A Locale in which to render the String.

Expires December 6, 2004

[Page 71]

2.15.8 toString

```
public String toString()
```

Overrides the default toString implementation. It expands all the nested exceptions.

2.15.9 Result codes

See [[LDAPPROTO](#)] for a discussion of the meanings and values of the codes. The corresponding ASN.1 name from [[LDAPPROTO](#)] is provided in parentheses. Applications should not use the result code to distinguish between server exceptions and local exceptions, but instead should use instanceof LDAPLocalException (see 2.18). All of the following are constants of LDAPException.

```
ADMIN_LIMIT_EXCEEDED (adminLimitExceeded)
AFFECTS_MULTIPLE_DSAS (affectsMultipleDSAs)
ALIAS_DEREFERENCING_PROBLEM (aliasDereferencingProblem)
ALIAS_PROBLEM (aliasProblem)
ATTRIBUTE_OR_VALUE_EXISTS (attributeOrValueExists)
AUTH_METHOD_NOT_SUPPORTED (authMethodNotSupported)
BUSY (busy)
COMPARE_FALSE (compareFalse)
COMPARE_TRUE (compareTrue)
CONFIDENTIALITY_REQUIRED (confidentialityRequired)
CONSTRAINT_VIOLATION (constraintViolation)
ENTRY_ALREADY_EXISTS (entryAlreadyExists)
INAPPROPRIATE_AUTHENTICATION (inappropriateAuthentication)
INAPPROPRIATE_MATCHING (inappropriateMatching)
INSUFFICIENT_ACCESS_RIGHTS (insufficientAccessRights)
INVALID_ATTRIBUTE_SYNTAX (invalidAttributeSyntax)
INVALID_CREDENTIALS (invalidCredentials)
INVALID_DN_SYNTAX (invalidDNSyntax)
IS_LEAF (isLeaf)
LOOP_DETECT (loopDetect)
NAMING_VIOLATION (namingViolation)
NO_SUCH_ATTRIBUTE (noSuchAttribute)
NO_SUCH_OBJECT (noSuchObject)
NOT_ALLOWED_ON_NONLEAF (notAllowedOnNonLeaf)
NOT_ALLOWED_ON_RDN (notAllowedOnRDN)
OBJECT_CLASS_MODS_PROHIBITED (objectClassModsProhibited)
OBJECT_CLASS_VIOLATION (objectClassViolation)
OPERATIONS_ERROR (operationsError)
OTHER (other)
PROTOCOL_ERROR (protocolError)
```

REFERRAL (referral)
SASL_BIND_IN_PROGRESS (saslBindInProgress)
SIZE_LIMIT_EXCEEDED (sizeLimitExceeded)

Expires December 6, 2004

[Page 72]


```
STRONG_AUTH_REQUIRED (strongAuthRequired)
SUCCESS (success)
TIME_LIMIT_EXCEEDED (timeLimitExceeded)
UNAVAILABLE (unavailable)
UNAVAILABLE_CRITICAL_EXTENSION (unavailableCriticalExtension)
UNDEFINED_ATTRIBUTE_TYPE (undefinedAttributeType)
UNWILLING_TO_PERFORM (unwillingToPerform)
```

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

```
AMBIGUOUS_RESPONSE (0x65)
AUTH_UNKNOWN (0x56)
CLIENT_LOOP (0x60)
CONNECT_ERROR (0x5b)
CONTROL_NOT_FOUND (0x5d)
DECODING_ERROR (0x54)
ENCODING_ERROR (0x53)
FILTER_ERROR (0x57)
INTERMEDIATE_RESPONSE (0x71)
INVALID_RESPONSE (0x64)
LDAP_NOT_SUPPORTED (0x5c)
LDAP_TIMEOUT (0x55)
LOCAL_ERROR (0x52)
MORE_RESULTS_TO_RETURN (0x5f)
NO_MEMORY (0x5a)
NO_RESULTS_RETURNED (0x5e)
REFERRAL_LIMIT_EXCEEDED (0x61)
SERVER_DOWN (0x51)
TLS_NOT_SUPPORTED (0x70)
UNKNOWN_TYPE (0x72)
USER_CANCELLED (0x58)
```

2.16 public class LDAPExtendedOperation **implements Cloneable, Serializable**

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

2.16.1 Constructors

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

Constructs a new object with the specified OID and data.

Parameters are:

oid The OID of the operation.

Expires December 6, 2004

[Page 73]

value	The BER-encoded operation-specific data of the operation. The API implementation does not interpret or convert the value.
-------	---

[2.16.2](#) **getID**

```
public String getID()
```

Returns the OID of the operation.

[2.16.3](#) **getValue**

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

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

[2.16.4](#) **setValue**

```
protected void setValue(byte[] value)
```

Sets the operation-specific data of the object. This method is for use by extensions of `LDAPExtendedOperation`.

Parameters are:

value	The BER-encoded operation-specific data of the operation to be assigned to the object (as a reference) The API implementation does not interpret or convert the value.
-------	--

[2.17](#) **public class LDAPExtendedResponse**

extends LDAPResponse implements Serializable

An `LDAPExtendedResponse` object encapsulates a server response to an extended operation request. Objects extending this class can be registered by OID (see 2.17.3), and are instantiated by the API implementation on receipt of an extended response with the given OID.

[2.17.1](#) **getID**

```
public String getID()
```

Returns the OID of the response.

[2.17.2](#) **getValue**

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

Expires December 6, 2004

[Page 74]

Returns the raw bytes of the value part of the response.

2.17.3 register

```
public static void register(String oid, Class extendedResponseClass)
```

Registers an application class to be instantiated on receipt of an extended response with the given oid. Any previous registration for the oid is overridden. The extendedResponseClass object MUST be an extension of LDAPExtendedResponse.

Parameters are:

oid The OID of the extended response

extendedResponseClass A class which can instantiate an LDAPExtendedResponse. The class must implement the following constructor signature:

```
public (String oid, byte[] value)
```

oid The OID of the extended response

value Response-specific data; the API implementation does not interpret or convert the value

2.18 public class LDAPLocalException extends LDAPException

This exception, derived from LDAPException, is thrown to report an LDAP error that does not originate from the server, and is not covered by other Exception classes such as IllegalArgumentException. For example, LDAPLocalException is thrown when network errors occur, when calling LDAPConnection.StartTLS() and operations are outstanding on the connection, or when a REFERRAL_LIMIT_EXCEEDED error occurs when following referrals.

2.18.1 Constructors

```
public LDAPLocalException()
```

Constructs a default exception with no specific error information.

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

Expires December 6, 2004

[Page 75]

Constructs an exception with a string describing the error and a result code.

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

Constructs an exception with a result code, a string describing the error, and an embedded root exception as additional information.

Parameters are:

message	The additional error information
resultCode	The result code returned
rootException	An exception which caused the failure, if any

[2.19](#) public class LDAPMatchingRuleSchema extends LDAPSchemaElement

The LDAPMatchingRuleSchema class represents the definition of a matching rule. It is used to query matching rule syntax, and to add or delete a matching rule definition in a Directory's subschema. See [\[ATTR\]](#) for a description of matching rule representation in LDAP.

[2.19.1](#) Constructors

```
public LDAPMatchingRuleSchema(String[] names,
                              String oid,
                              String description,
                              String[] attributes,
                              boolean obsolete,
                              String syntaxString)
```

Constructs a matching rule definition for adding to or deleting from a Directory's subschema. [\[LDAPPROTO\]](#) defines which parameters are optional (may be null).

```
public LDAPMatchingRuleSchema(String rawMatchingRule,
                              String rawMatchingRuleUse)
```

Constructs a matching rule definition from values encoded using the matchingRule syntax and the matchingRuleUse syntax [\[ATTR\]](#)

for the same rule.

Expires December 6, 2004

[Page 76]

Parameters are:

name	Name of the attribute.
oid	OID of the.
description	Optional description of the attribute.
attributes	OIDs of attributes to which the rule applies, or null if none.
obsolete	true if this matching rule is obsolete.
syntaxString	OID of the syntax that this matching rule is valid for.
rawMatchingRule	A matching rule definition encoded using the matchingRule syntax [ATTR].
rawMatchingRuleUse	A matching rule use definition encoded using the matchingRuleUse syntax [ATTR], or null if none.

[2.19.2](#) **getAttributes**

```
public String[] getAttributes()
```

Returns the OIDs of the attributes to which this rule applies.

[2.19.3](#) **getSyntaxString**

```
public String getSyntaxString()
```

Returns the OID of the syntax that this matching rule is valid for.

[2.20](#) **public class LDAPMatchingRuleUseSchema** **extends LDAPSchemaElement**

The LDAPMatchingRuleUseSchema class represents the definition of a matching rule use. It is used to discover or modify which attributes are suitable for use with an extensible matching rule. It contains the name and OID of a matching rule, and a list of attributes that it applies to. See [[ATTR](#)] for a description of matching rule use representation in LDAP.

[2.20.1](#) **Constructors**

```
public LDAPMatchingRuleUseSchema(String[] names,
```

Expires December 6, 2004

[Page 77]

```
String oid,  
String description,  
boolean obsolete,  
String[] attributes)
```

Constructs a matching rule use definition for adding to or deleting from the schema. [[LDAPPROTO](#)] defines which parameters are optional (may be null).

```
public LDAPMatchingRuleUseSchema(String raw)
```

Constructs a matching rule use definition from an encoding using the matchingRuleUse syntax [[ATTR](#)].

Parameters are:

names	Name(s) of the matching rule.
oid	OID of the matching rule.
description	Optional description of the matching rule use.
obsolete	true if the matching rule use is obsolete.
attributes	List of attributes that this matching rule applies to. These values may be either the names or OIDs of the attributes
raw	A matching rule use definition using the matchingRuleUse syntax [ATTR].

[2.20.2](#) **getAttributes**

```
public String[] getAttributes()
```

Returns an array of all the attributes that this matching rule applies to.

[2.21](#) **public class LDAPMessage** **implements Serializable**

Base class for asynchronous LDAP request and response messages and for LDAPExtendedResponse.

[2.21.1](#) **getControls**

```
public LDAPControl[] getControls()
```

Returns any controls in the message.

Expires December 6, 2004

[Page 78]

2.21.2 getMessageID

```
public int getMessageID()
```

Returns the message ID. Message IDs are defined in section 4.1.1.1 of [\[LDAPPROTO\]](#).

2.21.3 getType

```
public int getType()
```

Returns the LDAP operation type of the message. The type is one of the following:

```
BIND_REQUEST (0x0)
BIND_RESPONSE (0x1)
UNBIND_REQUEST (0x2)
SEARCH_REQUEST (0x3)
SEARCH_RESPONSE (0x4)
SEARCH_RESULT (0x5)
MODIFY_REQUEST (0x6)
MODIFY_RESPONSE (0x7)
ADD_REQUEST (0x8)
ADD_RESPONSE (0x9)
DEL_REQUEST (0xa)
DEL_RESPONSE (0xb)
MODIFY_RDN_REQUEST (0xc)
MODIFY_RDN_RESPONSE (0xd)
COMPARE_REQUEST (0xe)
COMPARE_RESPONSE (0xf)
ABANDON_REQUEST (0x10)
SEARCH_RESULT_REFERENCE (0x13)
EXTENDED_REQUEST (0x17)
EXTENDED_RESPONSE (0x18)
INTERMEDIATE_RESPONSE (0x19)
```

Each of the above types is a constant of LDAPMessage.

2.22 public class LDAPMessageQueue implements Serializable

Represents the message queue associated with a particular asynchronous LDAP operation or operations.

2.22.1 getMessageIDs

```
public int[] getMessageIDs()
```

Returns the message IDs for all outstanding requests, i.e. requests for which a response has not been received from the server or which

Expires December 6, 2004

[Page 79]

still have messages to be retrieved with `getResponse`. The last ID in the array is the `messageID` of the latest submitted request. Message IDs are defined in section 4.1.1.1 of [[LDAPPROTO](#)].

2.22.2 `getResponse`

`public LDAPMessage getResponse()` throws `LDAPException`

Blocks until a response is available, or until all operations associated with the object have completed or been canceled, and returns the response.

`public LDAPMessage getResponse(int msgid)` throws `LDAPException`

Blocks until a response is available for a particular message ID, or until all operations associated with the message ID have completed or been canceled, and returns the response. If there is no outstanding operation for the message ID (or if `msgid` is zero or a negative number), `IllegalArgumentException` is thrown.

Parameters are:

<code>msgid</code>	A particular message ID to query for responses available.
--------------------	---

2.22.3 `isComplete`

`public boolean isComplete(int msgid)`

Reports true if all results for a particular message ID have been received by the API implementation. For requests that return multiple results (for example search) there may still be messages queued in the object for retrieval by the application. If there is no outstanding operation for the message ID (or if `msgid` is zero or a negative number), `IllegalArgumentException` is thrown.

Parameters are:

<code>msgid</code>	A particular message ID to query for completion.
--------------------	--

2.22.4 `isResponseReceived`

`public boolean isResponseReceived()`

Reports true if any response has been received from the server and not yet retrieved with `getResponse`. If `getResponse` has been used to retrieve all messages received to this point, then `isResponseReceived`

returns false.

Expires December 6, 2004

[Page 80]


```
public boolean isResponseReceived(int msgid)
```

Reports true if a response has been received from the server for a particular message ID but not retrieved with `getResponse`. If there is no outstanding operation for the message ID (or if `msgid` is zero or a negative number), `IllegalArgumentException` is thrown.

Parameters are:

<code>msgid</code>	A particular message to query for responses available.
--------------------	--

[2.22.5](#) merge

```
public void merge(LDAPMessageQueue queue2)
```

Merges two queues. Moves/appends the content from the specified queue to this one. After the operation, `queue2.getMessageIDs()` returns an empty array and its outstanding responses have been removed (and appended to this queue).

[2.23](#) **public class LDAPModification** **implements Serializable**

Encapsulates a single change specification for an `LDAPAttribute`.

[2.23.1](#) Constructors

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

Specifies a modification to be made to an attribute.

Parameters are:

<code>op</code>	The type of modification to make, which can be one of the following:
<code>ADD</code>	The value should be added to the attribute, creating the attribute if necessary.
<code>DELETE</code>	The value should be removed from the attribute, removing the entire attribute if no values are listed, or if all

current values of the attribute
are listed for deletion.

Expires December 6, 2004

[Page 81]

REPLACE	The value should replace all existing values of the attribute with the new values listed, creating the attribute if it did not already exist. A replace with no value will delete the entire attribute if it exists, and is ignored if the attribute does not exist.
---------	--

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

[2.23.2](#) **getAttribute**

```
public LDAPAttribute getAttribute()
```

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

[2.23.3](#) **getOp**

```
public int getOp()
```

Returns the type of modification specified by this object.

[2.23.4](#) **Constants of LDAPModification**

ADD	(0)	The value should be added to the attribute, creating the attribute if necessary.
DELETE	(1)	The value should be removed from the attribute, removing the entire attribute if no values are listed, or if all current values of the attribute are listed for deletion.
REPLACE	(2)	The value should replace all existing values of the attribute with the new values listed, creating the attribute if it did not already exist. A replace with no value will delete the entire attribute if it exists, and is ignored if the attribute does not exist.

[2.24](#) **public class LDAPNameFormSchema extends LDAPSchemaElement**

The LDAPNameFormSchema class represents the definition of a Name Form. It is used to discover or modify the allowed naming attributes

Expires December 6, 2004

[Page 82]

for a particular object class. See [\[ATTR\]](#) for a description of name form representation in LDAP.

[2.24.1](#) Constructors

```
public LDAPNameFormSchema(String[] names,  
                           String oid,  
                           String description,  
                           boolean obsolete,  
                           String objectClass,  
                           String[] required,  
                           String[] optional)
```

Constructs a name form for adding to or deleting from the schema. [\[LDAPPROTO\]](#) defines which parameters are optional (may be null).

```
public LDAPNameFormSchema(String raw)
```

Constructs a DIT content rule from an encoding using the nameForms syntax [\[ATTR\]](#).

Parameters are:

names	Name(s) of the name form.
oid	OID of the name form.
description	Optional description of the name form.
obsolete	true if the name form is obsolete.
objectClass	The object to which this name form applies. This may either be specified by name or OID.
required	A list of the attributes that must be present in the RDN of an entry that this name form controls. These may either be specified by name or OID.
optional	A list of the attributes that may be present in the RDN of an entry that this name form controls. These may either be specified by name or OID.
raw	A name form definition encoded using the nameForms syntax [ATTR] .

[2.24.2](#) getObjectClass

Expires December 6, 2004

[Page 83]

```
public String getObjectClass()
```

Returns the name of the object class that this name form applies to.

[2.24.3](#) **getOptionalNamingAttributes**

```
public String[] getOptionalNamingAttributes()
```

Returns the list of optional naming attributes for an entry controlled by this content rule.

[2.24.4](#) **getRequiredNamingAttributes**

```
public String[] getRequiredNamingAttributes()
```

Returns the list of required naming attributes for an entry controlled by this name form.

[2.25](#) **public class LDAPObjectClassSchema** **extends LDAPSchemaElement**

The LDAPObjectClassSchema class represents the definition of an object class. It is used to query the syntax of an object class, and to add or delete an object class definition in a Directory's subschema. See [\[ATTR\]](#) for a description of object class representation in LDAP.

[2.25.1](#) **Constructors**

```
public LDAPObjectClassSchema(String[] names,  
                             String oid,  
                             String[] superiors,  
                             String description,  
                             String[] required,  
                             String[] optional,  
                             int type,  
                             boolean obsolete)
```

Constructs an object class definition for adding to or deleting from a Directory's subschema. [\[LDAPPROTO\]](#) defines which parameters are optional (i.e., which may be null).

```
public LDAPObjectClassSchema(String raw)
```

Constructs an object class definition from an encoding

using the `ObjectClassDescription` syntax [[ATTR](#)].

Expires December 6, 2004

[Page 84]

Parameters are:

names	Name(s) of the object class.
oid	OID of the object class.
description	Optional description of the object class.
superiors	The object classes this one derives from.
required	A list of attributes required for an entry with this object class.
optional	A list of attributes acceptable but not required for an entry with this object class.
type	One of ABSTRACT, AUXILIARY, or STRUCTURAL (See 2.25.6).
obsolete	true if this object class is obsolete.
raw	An object class definition encoded using the ObjectClassDescription syntax [ATTR].

[2.25.2](#) **getOptionalAttributes**

```
public String[] getOptionalAttributes()
```

Returns a list of attributes acceptable but not required of an entry with this object class.

[2.25.3](#) **getRequiredAttributes**

```
public String[] getRequiredAttributes()
```

Returns a list of attributes required of an entry with this object class.

[2.25.4](#) **getSuperiors**

```
public String[] getSuperiors()
```

Returns the object classes which this one derives from.

[2.25.5](#) **getType**

```
public int getType()
```

Expires December 6, 2004

[Page 85]

Returns one of ABSTRACT, AUXILIARY, or STRUCTURAL (See 2.25.6).

2.25.6 Constants of LDAPObjectClassSchema

ABSTRACT (0) identifies an abstract schema class
STRUCTURAL (1) identifies a structural schema class
AUXILIARY (2) identifies an auxiliary schema class

2.26 public class LDAPReferralException extends LDAPException

This exception, derived from LDAPException, is thrown when a server returns a referral or search reference on a synchronous request and when automatic referral following has not been enabled. The exception may also be thrown when automatic referral following is enabled, but only if there was an error while attempting to follow the referral.

2.26.1 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,  
                             Throwable rootException)
```

Constructs a default exception with a specified string as additional information and an exception that indicates a failure to follow a referral.

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

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

Parameters are:

message The additional error information.

Expires December 6, 2004

[Page 86]

<code>resultCode</code>	The result code returned
<code>serverMessage</code>	Error message specifying additional information from the server.
<code>rootException</code>	An exception which caused referral following to fail

[2.26.2](#) **getFailedReferral**

```
public String getFailedReferral()
```

Gets the referral URL that could not be followed. If multiple URLs are in the list, and none could be followed, the method returns one of them.

[2.26.3](#) **getReferrals**

```
public String[] getReferrals()
```

Gets the list of referral URLs (URLs to other servers) returned by the LDAP server. If the scope field of a referral of type `SearchResultReference` must be modified in order to follow the referral, the API implementation **MUST** modify the scope field of the URL before returning the URL to the application.

The referral list may include URLs of a type other than LDAP server (i.e. a referral URL other than `ldap://something`).

[2.26.4](#) **setFailedReferral**

```
public void setFailedReferral(String url)
```

Sets a referral URL that could not be followed.

[2.27](#) **public interface LDAPReferralHandler**

Shared ancestor to the two types of referral objects - `LDAPBindHandler` and `LDAPAuthHandler`.

[2.28](#) **public class LDAPResponse** **extends LDAPMessage**

Expires December 6, 2004

[Page 87]

Represents the response to a particular asynchronous LDAP operation. `LDAPExtendedResponse` extends `LDAPResponse` and is returned on a synchronous extended request.

[2.28.1](#) **getErrorMessage**

```
public String getErrorMessage()
```

Returns any error message in the response.

[2.28.2](#) **getMatchedDN**

```
public String getMatchedDN()
```

Returns the partially matched DN field, if any, in a server response.

[2.28.3](#) **getReferrals**

```
public String[] getReferrals()
```

Returns list of all referral URLs, if any, in a server response.

[2.28.4](#) **getResultCode**

```
public int getResultCode()
```

Returns the result code in a server response, as defined in [[LDAPPROTO](#)].

[2.29](#) **public class LDAPSchema** **extends LDAPEntry** implements Serializable

The `LDAPSchema` class provides methods to parse schema attributes associated with an `LDAPEntry`. Schema is retrieved from a Directory Server's subschema using the `fetchSchema` method of `LDAPConnection` (see 2.8.13).

[2.29.1](#) **Constructors**

```
public LDAPSchema(LDAPEntry entry)
```

Constructs an LDAPSchema object from attributes of an LDAPEntry. The object is empty if the entry parameter contains no schema attributes.

Expires December 6, 2004

[Page 88]

Parameters are:

entry	An LDAPEntry containing schema information.
-------	---

[2.29.2](#) **getAttributeNames**

```
public Enumeration getAttributeNames()
```

Returns an enumeration of attribute names.

[2.29.3](#) **getAttributeSchema**

```
public LDAPAttributeSchema getAttributeSchema( String name )
```

Returns a particular attribute definition, or null if not found.

Parameters are:

name	Name or OID of the attribute for which a definition is to be returned.
------	--

[2.29.4](#) **getAttributeSchemas**

```
public Enumeration getAttributeSchemas()
```

Returns an enumeration of attribute definitions.

[2.29.5](#) **getDITContentRuleNames**

```
public Enumeration getDITContentRuleNames()
```

Returns an enumeration of content rule names.

[2.29.6](#) **getDITContentRuleSchema**

```
public  
LDAPDITContentRuleSchema getDITContentRuleSchema(String name )  
Returns a particular content rule definition, or null if not found.
```

Parameters are:

name	Name or OID of the content rule for which a definition is to be returned.
------	---

Expires December 6, 2004

[Page 89]

[2.29.7](#) **getDITContentRuleSchemas**

```
public Enumeration getDITContentRuleSchemas()
```

Returns an enumeration of LDAPDITContentRuleSchema objects.

[2.29.8](#) **getDITStructureRuleNames**

```
public Enumeration getDITStructureRuleNames ()
```

Returns an enumeration of structure rule names.

[2.29.9](#) **getDITStructureRuleSchema**

```
public  
LDAPDITStructureRuleSchema getDITStructureRuleSchema( String name )  
  
public LDAPDITStructureRuleSchema getDITStructureRuleSchema( int id )
```

Returns a particular structure rule definition, or null if not found.

Parameters are:

name	Name or OID of the structure rule for which a definition is to be returned.
id	Identifier of the structure rule for which a definition is to be returned.

[2.29.10](#) **getDITStructureRuleSchemas**

```
public Enumeration getDITStructureRuleSchemas()
```

Returns an enumeration of LDAPDITStructureRuleSchema objects.

[2.29.11](#) **getMatchingRuleNames**

```
public Enumeration getMatchingRuleNames()
```

Returns an enumeration of matching rule names.

[2.29.12](#) **getMatchingRuleSchema**

```
public LDAPMatchingRuleSchema getMatchingRuleSchema( String name )
```

Returns a particular matching rule definition, or null if not found.

Expires December 6, 2004

[Page 90]

Parameters are:

name	Name or OID of the matching rule for which a definition is to be returned.
------	--

[2.29.13](#) **getMatchingRuleSchemas**

```
public Enumeration getMatchingRuleSchemas()
```

Returns an enumeration of matching rule definitions.

[2.29.14](#) **getMatchingRuleUseNames**

```
public Enumeration getMatchingRuleUseNames()
```

Returns an enumeration of matching rule use names.

[2.29.15](#) **getMatchingRuleUseSchema**

```
public  
LDAPMatchingRuleUseSchema getMatchingRuleUseSchema( String name )
```

Returns a particular matching rule use definition, or null if not found.

Parameters are:

name	Name or OID of the matching rule use for which a definition is to be returned.
------	--

[2.29.16](#) **getMatchingRuleUseSchemas**

```
public Enumeration getMatchingRuleUseSchemas()
```

Returns an enumeration of LDAPMatchingRuleUseSchema objects.

[2.29.17](#) **getNameFormNames**

```
public Enumeration getNameFormNames ( )
```

Returns an enumeration of name form names.

[2.29.18](#) **getNameFormSchema**

```
public LDAPNameFormSchema getNameFormSchema( String name )
```

Expires December 6, 2004

[Page 91]

Returns a particular name form definition, or null if not found.

Parameters are:

name	Name or OID of the name form for which a definition is to be returned.
------	--

[2.29.19](#) **getNameFormSchemas**

```
public Enumeration getNameFormSchemas()
```

Returns an enumeration of LDAPNameFormSchema objects.

[2.29.20](#) **getObjectClassNames**

```
public Enumeration getObjectClassNames()
```

Returns an enumeration of object class names.

[2.29.21](#) **getObjectClassSchema**

```
public LDAPObjectClassSchema getObjectClassSchema( String name )
```

Returns a particular object class definition, or null if not found.

Parameters are:

name	Name or OID of the object class for which a definition is to be returned.
------	---

[2.29.22](#) **getObjectClassSchemas**

```
public Enumeration getObjectClassSchemas()
```

Returns an enumeration of object class definitions.

[2.29.23](#) **getSyntaxSchema**

```
public LDAPSyntaxSchema getSyntaxSchema( String oid )
```

Returns a particular syntax definition, or null if not found.

Parameters are:

oid	OID of the syntax for which a definition is to be
-----	---

returned.

Expires December 6, 2004

[Page 92]

[2.29.24](#) **getSyntaxSchemas**

```
public Enumeration getSyntaxSchemas()
```

Returns an enumeration of LDAPSyntaxSchema objects.

[2.30](#) **public abstract class LDAPSchemaElement** **extends LDAPAttribute implements Serializable**

The LDAPSchemaElement class is the base class for representing schema elements in LDAP. All classes representing schema elements are read-only and thus do not support the addValue and removeValue methods from LDAPAttribute. This class overrides those methods and throws UnsupportedOperationException if addValue or removeValue is invoked.

[2.30.1](#) **getDescription**

```
public String getDescription()
```

Returns the description of the element. With respect to the protocol-level schema element syntax definition of [[ATTR](#)], the value is that of the DESC qualifier.

[2.30.2](#) **getNames**

```
public String[] getNames()
```

Returns the name(s) of the element.

[2.30.3](#) **getID**

```
public String getID()
```

Returns the OID of the element.

[2.30.4](#) **getQualifier**

```
public String[] getQualifier(String name)
```

Returns an array of all values of a qualifier of the element which is not defined in [[ATTR](#)]. This method may be used to access the values of vendor-specific qualifiers (which begin with "X-" [[ATTR](#)]).

Parameters are:

Expires December 6, 2004

[Page 93]

name The name of the qualifier, case-sensitive.

2.30.5 getQualifierNames

```
public Enumeration getQualifierNames()
```

Returns an enumeration of all qualifiers of the element which are not defined in [\[ATTR\]](#).

2.30.6 isObsolete

```
public boolean isObsolete()
```

Returns true if the element has the OBSOLETE qualifier in its LDAP definition [\[ATTR\]](#).

2.30.7 setQualifier

```
public void setQualifier(String name, String[] values)
```

Sets the values of a specified optional or experimental qualifier of the element. This method may be used to set the values of vendor-specific qualifiers (which begin with "X-" [\[ATTR\]](#)).

Parameters are:

name The name of the qualifier, case-sensitive.

values The values to set for the qualifier.

2.30.8 toString

```
public String toString()
```

Returns a String in a format suitable for directly adding to a Directory (defined in [\[ATTR\]](#), as a value of the particular schema element attribute. See the format definition for each derived class.

2.31 public class LDAPSearchConstraints extends LDAPConstraints

A set of options to control a search operation. There is always an LDAPSearchConstraints object associated with an LDAPConnection object; it can be changed with LDAPConnection.setConstraints, or

overridden by passing an LDAPSearchConstraints object to a search operation.

Expires December 6, 2004

[Page 94]

2.31.1 Constructors

```
public LDAPSearchConstraints()
```

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

```
public LDAPSearchConstraints(LDAPConstraints cons)
```

Constructs an LDAPSearchConstraints object initialized with values from an existing constraints object (LDAPConstraints or LDAPSearchConstraints).

```
public LDAPSearchConstraints(int msLimit,  
                             int serverTimeLimit,  
                             int dereference,  
                             int maxResults,  
                             boolean doReferrals,  
                             int batchSize,  
                             LDAPReferralHandler handler,  
                             int hop_limit)
```

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

Parameters are:

cons	Constraints object to use as template.
msLimit	Maximum time in milliseconds to wait for results The default of 0 means that there is no maximum time limit. This is an interface-enforced limit.
serverTimeLimit	Maximum time in seconds that the server should spend returning results. This is a server-enforced limit. The default of 0 means no time limit.
dereference	Specifies when aliases should be dereferenced. The value MUST be either Deref_NEVER, Deref_FINDING, Deref_SEARCHING, or Deref_ALWAYS (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

Expires December 6, 2004

[Page 95]

	if the server sends back a referral (false by default). It is ignored for asynchronous operations.
batchSize	Specify the number of results to block on during enumeration. 0 means to block until all results are in (1 by default). It is ignored for asynchronous operations.
handler	Custom authentication processor, called when the LDAPConnection needs to authenticate, typically on following a referral. The default of null specifies default authentication processing, i.e. anonymous authentication if doReferrals is true. The object implements either the LDAPBindHandler or the LDAPAuthHandler interface. It is ignored for asynchronous operations.
hop_limit	Maximum number of referrals to follow in a sequence when attempting to resolve a request, when doing automatic referral following. It is ignored for asynchronous operations. The value is 10 by default.

2.31.2 getBatchSize

```
public int getBatchSize()
```

Returns the blocking factor for synchronous searches. When retrieving results from the LDAPSearchResults object, a blocking factor of 0 indicates that the next() method blocks until all results are received from the server. A value of 1 indicates that next() returns each result when it is received. A value of 2 blocks until 2 results are received from the server or until the final result is received.

2.31.3 getDereference

```
public int getDereference()
```

Specifies when aliases should be dereferenced. Returns one of Deref_NEVER, Deref_FINDING, Deref_SEARCHING, or Deref_ALWAYS.

2.31.4 getMaxResults

```
public int getMaxResults()
```

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

Expires December 6, 2004

[Page 96]

[2.31.5](#) **getServerTimeLimit**

```
public int getServerTimeLimit()
```

Reports the maximum number of seconds that the server is to wait when returning search results while using this constraint object

[2.31.6](#) **setBatchSize**

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

Sets the blocking factor for synchronous searches. When retrieving results from the LDAPSearchResults object, a blocking factor of 0 indicates that the next() method blocks until all results are received from the server. A value of 1 indicates that next() returns each result when it is received. A value of 2 blocks until 2 results are received from the server or until the final result is received. The default is 1.

Parameters are:

batchSize	Blocking size on search enumerations.
-----------	---------------------------------------

[2.31.7](#) **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 Deref_NEVER, Deref_FINDING, Deref_SEARCHING, or Deref_ALWAYS.
-------------	--

[2.31.8](#) **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	Maximum number of search results to return.
------------	---

Expires December 6, 2004

[Page 97]

[2.31.9](#) **setServerTimeLimit**

```
public void setServerTimeLimit(int seconds)
```

Sets the maximum number of seconds that the server is to wait when returning search results. The parameter is only recognized on search operations. The default of 0 means no time limit.

[2.31.10](#) **Constants of LDAPSearchConstraints**

DEREF_NEVER (0) Aliases are never dereferenced.

DEREF_SEARCHING (1) Aliases are dereferenced when searching the entries beneath the starting point of the search (but not when finding the starting entry).

DEREF_FINDING (2) Aliases are dereferenced when finding the starting point for the search (but not when searching under that starting entry).

DEREF_ALWAYS (3) Aliases are always dereferenced (both when finding the starting point for the search and when searching under that starting entry).

[2.32](#) **public class LDAPSearchResult** **extends LDAPMessage**

An LDAPSearchResult object encapsulates a single asynchronous search result.

[2.32.1](#) **getEntry**

```
public LDAPEntry getEntry()
```

Returns the entry of a server search response.

[2.33](#) **public class LDAPSearchResultReference** **extends LDAPMessage**

An LDAPSearchResultReference object encapsulates a continuation reference from an asynchronous search operation.

[2.33.1](#) **getReferrals**

```
public String[] getReferrals()
```

Expires December 6, 2004

[Page 98]

Returns the list of any continuation reference URLs in the object.

[2.34](#) public class LDAPSearchResults

An LDAPSearchResults object is returned from a synchronous search operation. It provides access to all results received during the operation (entries and exceptions).

[2.34.1](#) getCount

```
public int getCount()
```

Returns a count of the entries and exceptions in the object. If the search was submitted with a batch size greater than 0, this reports the number of results received so far but not enumerated with next().

[2.34.2](#) getResponseControls

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

Returns the latest Server Controls returned by a Directory Server in the context of this search request.

[2.34.3](#) hasMore

```
public boolean hasMore()
```

Reports if there are more search results. If true, there are more search results.

[2.34.4](#) next

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

Returns the next search result as an LDAPEntry. If automatic referral following is disabled or a referral was not followed, next() will throw an LDAPReferralException when the referral is received. See also 2.31.6.

[2.35](#) public class LDAPSyntaxSchema extends LDAPSchemaElement

The LDAPSyntaxSchema class represents the definition of a syntax. It is used to discover the known set of syntaxes in effect for the

subschema. See [[ATTR](#)] for a description of syntax representation in LDAP.

Expires December 6, 2004

[Page 99]

Note, that though this extends `LDAPSchemaElement`, it does not use the `name` or `obsolete` members, subsequently calls to `getName` always return null and `isObsolete` always returns false. There is also no matching `getSyntaxNames` method in `LDAPSchema`.

Note also, that adding and removing syntaxes is not typically a supported feature of LDAP servers.

2.35.1 Constructors

```
public LDAPSyntaxSchema(String oid,  
                        String description)
```

Constructs a syntax for adding to or deleting from the schema.

```
public LDAPSyntaxSchema(String raw)
```

Constructs a syntax from an encoding using the `ldapSyntaxes` syntax [\[ATTR\]](#).

Parameters are:

<code>oid</code>	OID of the syntax.
<code>description</code>	Optional description of the syntax.
<code>raw</code>	A definition of a syntax encoded using the <code>ldapSyntaxes</code> syntax [ATTR] .

2.36 public interface LDAPUnsolicitedNotificationListener

An object that implements this interface can be notified when unsolicited messages arrive from the server. The Application registers the object with `LDAPConnection.addUnsolicitedNotificationListener`. Unsolicited messages have a message ID of 0. An implementation of the Java LDAP API SHOULD NOT generate messages with an ID of 0.

2.36.1 messageReceived

```
public void messageReceived(LDAPExtendedResponse msg)
```

The method is called when an unsolicited message arrives from a server, if the object has registered with `LDAPConnection.addUnsolicitedNotificationListener`.

Parameters are:

msg

An unsolicited message received from the server.

Expires December 6, 2004

[Page 100]

2.37 public class LDAPUrl **implements Cloneable, Serializable**

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

2.37.1 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 a URL object 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,  
               String[] extensions)
```

Constructs an LDAP URL with all fields explicitly assigned, to specify an LDAP search operation.

Parameters are:

url	An LDAP URL string, e.g. "ldap://ldap.example.com:80/dc=example,dc=com?cn, sn?sub?(objectclass=inetOrgPerson)".
host	Host identifier of LDAP server, or null for "localhost". See 2.8.9 for a discussion of valid identifiers.
port	Port number for LDAP server (use LDAPConnection.DEFAULT_PORT for default port).

dn Distinguished name of the base object of the
 search.

Expires December 6, 2004

[Page 101]

attrNames	Names or OIDs of attributes to retrieve. Passing a null array signifies that all user attributes are to be retrieved. Passing a value of "*" as an element of the array signifies that all user attributes are to be retrieved.
scope	Depth of search (in DN namespace). Use one of SCOPE_BASE, SCOPE_ONE, SCOPE_SUB from LDAPConnection.
filter	Search filter specifying the search criteria, as defined in [FILTER].
extensions	LDAP URL extensions specified; may be null or empty. Each extension is a type=value expression. The =value part can be omitted. Prefix the expression with '!' if the extension is mandatory for evaluation of the URL.

[2.37.2](#) decode

```
public static String decode(String URLEncoded)
                        throws MalformedURLException
```

Decodes a URL-encoded string. Any occurrences of %HH are decoded to the hex value represented. However, this method does NOT decode "+" into " ". See [[URL](#)] for details on URL encoding/decoding.

Parameters are:

URLEncoded String to decode.

[2.37.3](#) encode

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

Encodes the specified string. Any illegal characters are encoded as %HH.

Parameters are:

toEncode String to encode.

[2.37.4](#) getAttributeArray

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

Returns an array of attribute names specified in the URL.

Expires December 6, 2004

[Page 102]

2.37.5 getAttributes

```
public Enumeration getAttributes()
```

Returns an enumerator for the attribute names specified in the URL. The enumerator is empty if the URL contains no attribute names.

2.37.6 getDN

```
public String getDN()
```

Returns the distinguished name encapsulated in the URL, or null if none is specified.

2.37.7 getExtensions

```
public String[] getExtensions ()
```

Returns any LDAP URL extensions specified, or null if none are specified. Each extension is a type=value expression. The =value part can be omitted. Prefix the expression with '!' if the extension is mandatory for evaluation of the URL.

2.37.8 getFilter

```
public String getFilter()
```

Returns the search filter [[LDAPURL](#)], or null if none was specified.

2.37.9 getHost

```
public String getHost()
```

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

2.37.10 getPort

```
public int getPort()
```

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

2.37.11 getScope

```
public int getScope()
```

Expires December 6, 2004

[Page 103]

Returns the depth of search (in DN namespace) - one of SCOPE_BASE, SCOPE_ONE, SCOPE_SUB from LDAPConnection.

[2.37.12](#) toString

```
public String toString()
```

Returns a valid string representation of this LDAP URL.

[3.](#) Implementation considerations

[3.1](#) Controls

LDAPv3 operations can be extended through the use of controls. Controls can be sent to a server or returned to the application with any LDAP message. These controls are represented by LDAPControl objects.

Controls are set and retrieved in LDAPConnection with the setConstraints and getConstraints methods. Either a single LDAPControl or an array can be specified, e.g.

```
LDAPControl control = new LDAPControl( type, critical, vals );
LDAPConstraints cons = ld.getConstraints();
cons.setControls( control );
ld.setConstraints( cons );
or
LDAPControl[] controls = new LDAPControl[2];
controls[0] = new LDAPControl( type0, critical0, vals0 );
controls[1] = new LDAPControl( type1, critical1, vals1 );
LDAPConstraints cons = ld.getConstraints();
cons.setControls( controls );
ld.setConstraints( cons );
```

Server controls returned to an application as part of the response to a synchronous operation can be obtained with LDAPConnection.getResponseControls() or LDAPSearchResults.getResponseControls(). Controls returned on an asynchronous operation are available with LDAPMessage.getControls().

[3.2](#) Referral handling and exceptions

Asynchronous requests

No automatic referral following is supported for asynchronous

requests.

Expires December 6, 2004

[Page 104]

No `LDAPExceptions` are thrown for asynchronous requests except in the case of `LDAPLocalException`. The `Throwable` causing the local error can be retrieved with `LDAPConnection.getCause()`.

When a referral is received, the application receives an `LDAPResponse` object with a result code of `REFERRAL`. If a continuation reference is received the application receives an `LDAPSearchResultReference` object.

Synchronous requests

Referral-following behavior depends on two things: if automatic referral following is enabled, and when enabled if an `LDAPReferralHandler` is provided by the application.

- Behavior if automatic referral following is "false" (not enabled):

An `LDAPException` is thrown for any non-zero result code.

An `LDAPException` is thrown for local failures.

An `LDAPReferralException` is thrown if the result code is `REFERRAL`. The object contains the referral URL strings. This ends the request, i.e., `LDAPSearchResults.hasMore()` will return false since there are no results to retrieve.

An `LDAPReferralException` is thrown for each `SearchResultReference` received during a synchronous search request. The object contains the referral URL strings. The exception is not an error, `LDAPSearchResults.hasMore()` will indicate if there are more results to retrieve.

- Behavior if automatic referral following is "true" and either no `LDAPReferralHandler` is registered in `LDAPConstraints`, or an `LDAPAuthHandler` object is registered in `LDAPConstraints`:

Exception handling is the same whether or not an `LDAPAuthHandler` object is registered, but authentication to the server is not the same. The registration of an `LDAPAuthHandler` object allows connections to be authenticated. No handler registered results in an anonymous (unauthenticated) connection.

`LDAPExceptions` are thrown if errors occur during the normal processing of the command, i.e. `NO_SUCH_OBJECT`, `NO_SUCH_ATTRIBUTE`, etc.

An `LDAPReferralException` is thrown only if the API implementation could not connect and bind to any referral URL in

the list of URLs included in an individual REFERRAL result or
LDAPSearchResultReference.

Expires December 6, 2004

[Page 105]

When problems occur during the establishment of an authenticated connection by the API implementation during automatic referral following, the `LDAPReferralException` will contain the last tried URL String of the server the API attempted to connect/bind to (the referral can be retrieved with `getFailedReferral()`), and the `Throwable` that caused the Exception (which can be retrieved with `getCause()`). Some possible failures are `IOException` on the connection, `MalformedURLException` on the referral string, or authentication failures on the bind.

Upon receipt of an `LDAPReferralException`, the application knows that the API implementation was not able to connect to any of the servers in the referral list (which can be retrieved with `getReferrals()`) and was not able to follow the referral. The error indicated by `getCause()` occurred on the referral indicated with `getFailedReferral()`. In the case of search, any results starting at the indicated base of the referred to server are missing from the search results.

- behavior if automatic referral following is "true", and an `LDAPBindHandler` object is registered in `LDAPConstraints`:

`LDAPExceptions` are thrown if errors occur during the normal processing of the command, i.e. `NO_SUCH_OBJECT`, `NO_SUCH_ATTRIBUTE`, etc.

An `LDAPReferralException` is thrown only if the `LDAPBindHandler` object throws an exception.

Upon receipt of an `LDAPReferralException`, the application knows that the `LDAPBindHandler` object was not able or not willing to connect to any of the servers in the referral list and so was not able to follow the referral. The list of referrals that failed is available with `getReferrals()` and the last failed referral with `getFailedReferral()`. The exception thrown by the `LDAPBindHandler` object is available with `getCause()`.

The API implementation MAY follow referrals of types other than LDAP URLs [[LDAPURL](#)] on automatic referral following if access to the referred servers is through the LDAP protocol [[LDAPv3](#)]. To successfully follow such referral URLs, the application MUST provide an `LDAPBindHandler` that can interpret the URL and perform an appropriate connect and bind operation to the server. The application MAY follow such referrals through application specific code.

[3.3](#) Message IDs

An implementation of the Java LDAP API SHOULD NOT generate messages with an ID of 0.

3.4 Notice of disconnection

Expires December 6, 2004

[Page 106]

If a notice of disconnection is received by a connection object, the API implementation **MUST** close the connection without accepting or sending additional messages. Any clones of the object are disconnected as a consequence of closing the connection.

[3.5](#) Level of compatibility

Implementations of the API **MUST** include all classes and interfaces described in this document and thus are to be binary compatible, i.e. any application with its classpath including the jar or class files implementing this API will exhibit consistent behavior when using the API as defined by this document.

[3.6](#) Dependencies

The Java LDAP API is dependent on the following:

- JDK 1.2 or higher
- JAAS 1.0 or higher (for the interfaces in the `javax.security.auth.callback` package)

[3.7](#) Invalid responses

If a message is received by an API implementation from the server and the message cannot be interpreted as an LDAP PDU, an `LDAPException` **MUST** be thrown with a result code of `INVALID_RESPONSE`.

[3.8](#) Java Unicode Limitations

Any Unicode characters which cannot be represented with Java 16-bit Unicode strings (UCS2) cannot be used with this API, unless those characters are handled as binary UTF-8 data. If changes are introduced into the Java Language to accommodate these characters, implementations of the Java LDAP API **SHOULD** also accommodate these characters.

[3.9](#) Intermediate Response Messages

Applications may receive messages with a response type of `LDAP_INTERMEDIATE_RESPONSE` from any operation [[LDAPINT](#)].

Applications using synchronous interfaces will receive an `INTERMEDIATE_RESPONSE` exception upon receipt of an LDAP Intermediate Response Message; the operation will not be terminated, but will continue until an `LDAPResponse` message is received from the server or

until the application abandons the operation. The application can obtain the actual message through the `getResponse()` method of the `LDAPException` class.

Applications using asynchronous interfaces receive an `LDAPMessage` class with the response type set to `LDAP_INTERMEDIATE_RESPONSE`.

It is up to the application to determine how to interpret the data in the message.

3.10 Extensibility

To accommodate additions to the LDAP protocol [[LDAPINT](#)], LDAP messages with types not defined in [[LDAPPROTO](#)] MUST be returned to the application. Responses with unknown types that do not match the Message ID of an outstanding request or that are not an unsolicited notification message, SHOULD be discarded.

Applications using synchronous interfaces will receive an `UNKNOWN_TYPE` exception upon receipt of an LDAP message containing an unknown response type; the operation will not be terminated but will continue until an `LDAPResponse` message is received from the server or until the application abandons the operation. The application can obtain the actual message through the `getResponse()` method of the `LDAPException` class.

Applications using asynchronous interfaces will receive an `LDAPMessage` class with the response type set to the type of the message received.

It is up to the application to determine how to interpret the data in the message.

4. 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.

If a TLS session is terminated by the server but the client TLS provider and LDAP API implementation continue to use the socket rather than closing it, the application is notified through an `LDAPException` on the first operation request subsequent to termination of the TLS session.

An interface to using SASL for configurable authentication and session protection is provided, but implementations are outside the scope of this document. Implementations of this API MUST ensure that a SASL provider is configured to comply with the minimal security

guidelines of [RFC 2829](#) [[AUTH](#)].

Implementations of this API SHOULD be cautious when handling

Expires December 6, 2004

[Page 108]

authentication credentials. In particular, keeping long-lived copies of credentials without the application's knowledge is discouraged.

Implementations of this API MUST discard information about the server obtained prior to negotiation of security protections provided by SASL and/or TLS [[AUTH](#)].

5. Acknowledgements

The proposed API builds on earlier work done in collaboration with Thomas Kwan and Stephan Gudmundson, then of NCware Technologies Corp. It also includes suggestions by Steven Merrill of Novell, Inc, and benefited from extensive review and comments by Kurt Zeilenga of OpenLDAP, Rosanna Lee of Sun Microsystems, Mark Smith of Netscape Communications Corp., and Jim Sermersheim of Novell, Inc. Parts of the overview of the LDAP model are taken from [draft-ietf-ldapext-ldap-c-api](#).

6. Bibliography

6.1 Normative References

- [ATTR] M. Wahl, A. Coulbeck, T. Howes, S. Kille, "Lightweight Directory Access Protocol: Attribute Syntax Definitions", [RFC 2252](#), December 1997
- [AUTH] M. Wahl, H. Alvestrand, J. Hodges, R. Morgan, "Authentication Methods for LDAP", [RFC 2829](#), May 2000
- [DN] S. Kille, " UTF-8 String Representation of Distinguished Names," [RFC 2253](#), December 1997.
- [FILTER] T. Howes, "A String Representation of LDAP Search Filters," [RFC 2254](#), December 1997.
- [LDAPINT] R. Harrison, K. Zeilenga, "The Lightweight Directory Access Protocol (LDAP) Intermediate Response Message", [RFC 3771](#), April 2004
- [LDAPLANG] M. Wahl, T. Howes, "Use of Language Codes in LDAP", [RFC 2596](#), May 1999
- [LDAPPROTO] J. Hodges & R. Morgan, "LDAPv3: Technical Specification", [RFC 3377](#), September 2002
- [LDAPTLS] J. Hodges, R. Morgan, M. Wahl, "Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security",

[RFC 2830](#), May 2000.

Expires December 6, 2004

[Page 109]

- [LDAPURL] T. Howes, M. Smith, "An LDAP URL Format", [RFC 2255](#), December 1997.
- [LDAPv3] M. Wahl, T. Howes, S. Kille, "Lightweight Directory Access Protocol (v3)", [RFC 2251](#), December 1997.
- [SASL] J. Myers, "Simple Authentication and Security Layer (SASL)", [RFC 2222](#), October 1997.
- [TLS] T. Dierks, C. Allen, "The TLS Protocol", [RFC 2246](#), January 1999.
- [URL] T. Berners-Lee, R. Fielding, L. Masinter, " Uniform Resource Identifiers (URI): Generic Syntax", [RFC 2396](#), August 1998.

6.2 Informative References

- [IPv6] R. Hinden, S. Deering, "IP Version 6 Addressing Architecture", [RFC 2373](#), July 1998
- [IPv6URL] R. Hinden, B. Carpenter, L. Masinter, "Format for Literal IPv6 Addresses in URL's", [RFC 2732](#), December 1999
- [JAVA] B. Joy, G. Steele, J. Gosling, G. Bracha, "The Java Language Specification", Second Edition, Addison-Wesley, June 2000
- [JAVASASL] "Java SASL Specification", Java Community Process, JSR28
- [KEYWORDS] "Key words for use in RFCs to Indicate Requirement Levels", Bradner, S., [RFC 2119](#), March 1997
- [LANG] H. Alvestrans, "Tags for the Identification of Languages", [RFC 3066](#), January 2001.
- [X500] The Directory: Overview of Concepts, Models, and Services. CCITT, Recommendation X.500, 2nd edition, 1993

7. Authors' addresses

Rob Weltman
Netscape Communications Corp.
466 Ellis Street
Mountain View, CA 94043
USA
+1 650 937-3194
rweltman@netscape.com

Christine Tomlinson
Sun Microsystems, Inc.
8911 Capital of Texas Highway

Expires December 6, 2004

[Page 110]

Suite 4140
Austin, TX US 78759
+1 512 231 1600
christine.tomlinson@sun.com

Steven Sonntag
Novell, Inc.
1800 South Novell Place
Provo, UT 84606
USA
+1 801 861 7097
vtag@novell.com

Expires December 6, 2004

[Page 111]

8. Appendix A - Sample Java LDAP programs

8.1 Java LDAP programs using synchronous methods

```
import org.ietf.ldap.*;
import java.util.*;

public class SearchJensen {
    public static void main( String[] args ) {
        LDAPConnection ld = new LDAPConnection();
        try {
            /* Connect to server */
            String MY_HOST = "localhost";
            int MY_PORT = LDAPConnection.DEFAULT_PORT;
            ld.connect( MY_HOST, MY_PORT );
            /* Authentication not required for an anonymous
               connection */

            /* search for all entries with surname of Jensen */
            String MY_FILTER = "(sn=Jensen)";
            String MY_SEARCHBASE = "dc=example,dc=com";

            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 );
            ld.setSearchConstraints(cons);
            LDAPSearchResults res = ld.search( MY_SEARCHBASE,
                                              ld.SCOPE_ONE,
                                              MY_FILTER,
                                              null,
                                              false,
                                              cons );

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

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

                /* Get the attributes of the entry */
                LDAPAttributeSet findAttrs =
                    findEntry.getAttributeSet();
                Iterator enumAttrs = findAttrs.iterator();
            }
        }
    }
}
```

```
System.out.println( "Attributes: " );  
/* Loop on attributes */  
while ( enumAttrs.hasNext() ) {
```

Expires December 6, 2004

[Page 112]


```
LDAPAttribute anAttr =
    (LDAPAttribute)enumAttrs.next();
String attrName = anAttr.getName();
System.out.println( " " + attrName );
/* Loop on values for this attribute.
   Note: we are assuming all values are UTF-8
   strings
*/
Enumeration enumVals = anAttr.getStringValues();
while ( enumVals.hasMoreElements() ) {
    String aVal = (String)enumVals.nextElement();
    System.out.println( " " + aVal );
}
}

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

Expires December 6, 2004

[Page 113]

```
import org.ietf.ldap.*;
import java.io.*;
import java.util.*;
import javax.security.auth.callback.*;

public class ModifyEmail {
    public static void main( String[] args ) {
        LDAPConnection ld = new LDAPConnection();
        try {
            /* Connect to server */
            String MY_HOST = "localhost";
            int MY_PORT = LDAPConnection.DEFAULT_PORT;
            ld.connect( MY_HOST, MY_PORT );
            String MY_NAME =
                "uid=bjensen,ou=people,dc=example,dc=com";

            /* Callback handler to supply credentials for SASL */
            CallbackHandler cbh = new CallbackHandler() {
                public void handle( Callback[] callbacks )
                    throws IOException, UnsupportedCallbackException {
                    for ( int i = 0; i < callbacks.length; i++ ) {
                        if ( callbacks[i] instanceof NameCallback ) {
                            ((NameCallback)callbacks[i]).setName(
                                "bjensen" );
                        } else if ( callbacks[i] instanceof
                            PasswordCallback ) {
                            ((PasswordCallback)callbacks[i]).setPassword(
                                "MysteryLady".toCharArray() );
                        } else {
                            throw new UnsupportedCallbackException (
                                callbacks[i],
                                "Unrecognized Callback" );
                        }
                    }
                }
            };

            /* SASL bind */
            ld.bind( "cn=Barbara Jensen,dc=example,dc=com",
                "bjensen", null, cbh );

            /* Prepare to change my email address */
            LDAPAttribute attrEmail =
                new LDAPAttribute( "mail", "babs@example.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" );
```

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



```
import org.ietf.ldap.*;
import java.util.*;

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

            /* Fetch the schema */
            LDAPSchemas schema = ld.fetchSchema( ld.getSchemaDN() );

            /* What is the definition of "userPassword"? */
            LDAPAttributeSchema a =
                schema.getAttributeSchema( "userpassword" );
            if ( a != null ) {
                String syntax = a.getSyntaxString();
                String syntaxString = "string";
                if ( syntax.equals(BINARY_SYNTAX) )
                    syntaxString = "binary";
                else if ( syntax.equals(INTEGER_SYNTAX) )
                    syntaxString = "integer";
                String single = "multi-valued";
                if ( a.isSingleValued() )
                    single = "single-valued";
                System.out.println( "userPassword. " +
                                    "OID = " + a.getID() +
                                    ", type = " + syntaxString +
                                    ", " + single );
            }

            /* What are the possible attributes for "person"? */
            LDAPObjectClassSchema o =
                schema.getObjectClassSchema( "person" );
            if ( o != null ) {
                Enumeration required = o.getRequiredAttributes();
                Enumeration optional = o.getOptionalAttributes();
                System.out.println(
                    "Required attributes for person:" );
                while( required.hasMoreElements() ) {
                    System.out.println( " " +
                                            required.nextElement() );
                }
                System.out.println(
                    "Optional attributes for person:" );
            }
        }
    }
}
```

```
while( optional.hasMoreElements() ) {  
    System.out.println( " " +  
        optional.nextElement() );  
}
```

Expires December 6, 2004

[Page 116]


```
    }

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

protected static final String BINARY_SYNTAX =
    "1.3.6.1.4.1.1466.115.121.1.5";
protected static final String INTEGER_SYNTAX =
    "1.3.6.1.4.1.1466.115.121.1.27";
}
```

Expires December 6, 2004

[Page 117]

8.2 Java LDAP programs using asynchronous methods

```
import org.ietf ldap.*;
import java.util.*;
import java.io.UnsupportedEncodingException;

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;
            byte[] MY_PASSWORD = null;
            try {
                MY_PASSWORD = "password".getBytes( "UTF8" );
            } catch (UnsupportedEncodingException ex ) {
            }
            ld.connect( MY_HOST, MY_PORT );

            /* Asynchronous authentication */
            LDAPMessageQueue r =
                ld.bind( 3, "uid=admin,ou=people,dc=example,dc=com",
                        MY_PASSWORD, (LDAPMessageQueue)null );

            /* Do something else, just to show that we're not
               blocked yet */
            System.out.println( "Started authenticating" );

            /* Wait until it completes */
            LDAPResponse response = (LDAPResponse)r.getResponse();
            int resultCode = response.getResultCode();
            if (resultCode != LDAPEXCEPTION.SUCCESS) {
                throw new LDAPException ( "error result",
                                           resultCode,
                                           response.getMatchedDN() );
            }

            /* search for all entries with surname of Jensen */
            String MY_FILTER = "(sn=Jensen)";
            String MY_SEARCHBASE = "dc=example,dc=com";

            LDAPMessageQueue l =
                ld.search( MY_SEARCHBASE,
                          ld.SCOPE_ONE,
                          MY_FILTER,
                          null,
                          false,
```

```
(LDAPMessageQueue)null );
```

```
/* Loop on results until finished */  
LDAPMessage msg;
```

Expires December 6, 2004

[Page 118]

```
while( (msg = l.getResponse()) != null ) {
    if ( msg instanceof LDAPSearchResultReference ) {
        String[] urls =
            ((LDAPSearchResultReference)msg).getReferrals();
        // Do something with the referrals...
    } else if ( msg instanceof LDAPSearchResult ) {
        LDAPEntry entry =
            ((LDAPSearchResult)msg).getEntry();
        // The rest of the processing is the same as for
        // a synchronous search
        System.out.println( entry.getDN() );
    } else {
        // A search response
        LDAPResponse res = (LDAPResponse)msg;
        int status = res.getResultCode();
        if ( status == LDAPException.SUCCESS ) {
            // Nothing to do
        } else {
            String err =
                LDAPException.resultCodeToString(status);
            throw new LDAPException(err,
                                    status,
                                    res.getMatchedDN());
        }
    }
}

/* Done, so disconnect */
ld.disconnect();
} catch ( LDAPException e ) {
    System.err.println( e.toString() );
}
}
```

Expires December 6, 2004

[Page 119]

```
import org.ietf.ldap.*;
import java.util.*;

/* This example multiplexes the input from three different servers */

public class MultiplexServers {
    public static void main( String[] args ) {
        try {
            LDAPConnection [] ld = new LDAPConnection[3];
            String[] hosts = { "foo1", "foo2", "foo3" };
            int[] ports = { 389, 389, 2018 };
            String[] bases = { "dc=example,dc=com",
                               "o=example.com",
                               "dc=example2,dc=com" };
            /* search for all entries with surname of Jensen */
            String MY_FILTER = "(sn=Jensen)";

            for( int i = 0; i < ld.length; i++ ) {
                ld[i] = new LDAPConnection();
                /* Connect to server */
                ld[i].connect( hosts[i], ports[i] );
            }

            /* Get a response queue for one search */
            LDAPMessageQueue l =
                ld[0].search( bases[0],
                             ld.SCOPE_SUB,
                             MY_FILTER,
                             null,
                             false,
                             (LDAPMessageQueue)null );
            /* Share the queue */
            for( int i = 1; i < ld.length; i++ ) {
                ld[i].search( bases[i],
                             ld[i].SCOPE_SUB,
                             MY_FILTER,
                             null,
                             false,
                             l );
            }

            /* Loop on results until finished */
            LDAPMessage msg;
            while( (msg = l.getResponse()) != null ) {
                /* The rest is the same as in the previous example */
                /* ... */
            }
        }
    }
}
```

Expires December 6, 2004

[Page 120]


```
import org.ietf.ldap.*;
import java.util.*;

/* This example multiplexes the input from three searches in
   different subtrees of the same server */

public class MultiplexTrees {
    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 );
            String MY_FILTER = "(sn=Jensen)";
            String[] bases = { "dc=example,dc=com",
                               "o=example.com",
                               "dc=example2,dc=com" };

            /* Get a response queue for one search */
            LDAPMessageQueue l =
                ld.search( bases[0],
                           ld.SCOPE_SUB,
                           MY_FILTER,
                           null,
                           false,
                           (LDAPMessageQueue)null );

            /* Share the queue */
            for( int i = 1; i < bases.length; i++ ) {
                ld.search( bases[i],
                           ld.SCOPE_SUB,
                           MY_FILTER,
                           null,
                           false,
                           l );
            }

            /* The rest is the same as in the MultiplexServers
               example */
            /* ... */
        }
    }
}
```

Expires December 6, 2004

[Page 121]

```
import org.ietf.ldap.*;
import java.util.*;

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

            /* Use TLS to authenticate and secure the connection */
            ld.startTLS();
            /* Use SASL external on completed TLS client auth */
            ld.bind( null, null, new String[] { "external" },
                null, null );

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

            /* Now modify the entry in the directory */
            LDAPMessageQueue r =
                ld.modify( MY_NAME, mod, null );

            /* Do something else, just to show that we're not
               blocked yet */
            System.out.println( "Started authenticating" );

            /* Wait until it completes */
            LDAPResponse response = (LDAPResponse)r.getResponse();
            int resultCode = response.getResultCode();
            if (resultCode != LDAPException.SUCCESS) {
                throw new LDAPException ( "error result",
                    resultCode,
                    response.getMatchedDN() );
            }

            System.out.println( "Entry modified" );

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

}

}

Expires December 6, 2004

[Page 122]

```
}
```

Expires December 6, 2004

[Page 123]

[9. Appendix B](#) - Revision history

[9.1](#) Changes from ldap-java-api-18.txt

Updated the Status of this Memo and Copyright sections per [RFC 3668](#).

Made usage of MUST, MAY, SHOULD conforms to [RFC 2119](#).

Made editorial changes.

Clarified definition of OID.

Made a distinction between the API implementation and the application using the implementation.

LDAPAttribute

Clarified that name and options are case insensitive and options have no order.

Renamed method getBaseName to getTypeName.

Removed method getLangSubtype.

Renamed method getSubtypes to getOptions.

Renamed method hasSubtype to hasOption.

Renamed method hasSubtypes to hasOptions.

Changed terminology throughout the document from subtype to option and removed references to language subtypes, leaving only a general discussion about options.

LDAPAttributeSchema

Clarified names and values of constants.

LDAPAttributeSet

Clarified that method getAttribute returns null if no attribute found.

Removed method getAttribute(String, String) because it was associated with language subtypes.

Added method getSubset(String, String) which provides a more general implementation of the above.

LDAPConnection

Expires December 6, 2004

[Page 124]

Clarified that the method `compare` throws an `IllegalArgumentException` if more than one assertion value is supplied in the parameters.

Clarified which `bind()` methods are synchronous and which are asynchronous.

LDAPConstraints

Clarified that the method `getControls` returns null if no controls are present.

LDAPException

Clarified use of `LDAPLocalException` to distinguish local errors.

Placed methods in alphabetical order.

Placed local errors in alphabetical order.

Added method `getResponse` which is used to obtain message of unknown type of type `LDAP_INTERMEDIATE_RESPONSE`.

LDAPExtendedResponse

Implements `Serializable`

LDAPLocalException

Clarified use of `LDAPLocalException` to distinguish local errors.

LDAPMessage

Added a new response type: `LDAP_INTERMEDIATE_RESPONSE`

LDAPMessageQueue

Now a class instead of an interface. `LDAPResponseQueue` and `LDAPSearchQueue` are removed. If an application using asynchronous methods merged a search queue into a response queue, the search queue functionality of the `isComplete` method is needed but not present in the response queue and the identity of the queues is muddled. If you add the method, both classes will have the same functionality. There is really no need for the two classes, as the single class `LDAPMessageQueue` can perform all the needed functionality with no ambiguity.

LDAPResponseQueue

Removed class, only LDAPMessageQueue is needed.

LDAPSchemaElement

Expires December 6, 2004

[Page 125]

Implements Serializable

LDAPSearchQueue

Removed class, only LDAPMessageQueue is needed.

LDAPSchemaElement

Implements Serializable.

LDAPSocketFactory

Remove class, now using standard Java JSSE, such as
SSLSocketFactory.

LDAPTLSocketFactory

Remove class, now using standard Java JSSE.

Implementation Notes

Added a note describing what action is taken for referrals that are not a URL of type ldap://.

Moved the note regarding limitations of Java with regard to support for UCS4 characters in a String from the overview to the Implementation Notes and included notes on how to handle those characters.

Added information on how receipt of an LDAP Intermediate Response Message is handled.

Added information on how receipt of an LDAP message with an unknown response type is handled.

Bibliography

Moved some references from Informative to Normative and added [RFC 3771](#), LDAP Intermediate Response Message.

9.2 Changes from ldap-java-api-17.txt

LDAPAttribute

Throws IllegalArgumentException for null as value parameter in constructors.

Implements Comparable, with the method compareTo().

Expires December 6, 2004

[Page 126]

LDAPConnection

getProperty() returns null for property not found rather than throwing LDAPException.
Added fetchSchema() and getSchemaDN().
Removed getInputStream(), getOutputStream(), setInputStream(), setOutputStream().

LDAPConstraints

Removed text saying that non-LDAP URLs are ignored.
setProperty() throws IllegalArgumentException for an unsupported property.

LDAPEntry

Implements Comparable, with the method compareTo().
Clarified that getAttributeSet() and getAttribute() return copies rather than references.

LDAPExtendedOperation

Implements Cloneable.

LDAPExtendedResponse

Added register().

LDAPLocalException

New class for local errors.

LDAPSchema

It now extends LDAPEntry.
Constructor takes an LDAPEntry as parameter.
Removed fetchSchema(), add(), modify(), remove(), and saveSchema(). Methods in LDAPConnection provide this functionality.

LDAPSchemaElement

Extends LDAPAttribute.

Added clarification that the class is read only.

Expires December 6, 2004

[Page 127]

LDAPTLSSocketFactory

New interface to indicate ability to create a TLS socket.

Other

Java 2 is now a requirement (not JDK 1.1.8 or Java 2).
Editorial changes

[9.3](#) Changes from ldap-java-api-16.txt

LDAPException

Added serverMessage as parameter in constructors.

Other

Corrected typographical errors and errors in examples of [appendix A](#).

[9.4](#) Changes from ldap-java-api-15.txt

LDAPAttribute

Implements Cloneable.

LDAPAttributeSchema

Constructor takes "String[] names" instead of "String name" and "String[] aliases".

LDAPAttributeSet

Implements Cloneable and java.util.Set. Removed the methods made redundant by implementing Set: add(LDAPAttribute attr), elementAt(), getAttributes(), remove(String name), removeElementAt(), and size().

LDAPConnection

Removed all bind() signatures which take String instead of
byte[] for password.

Expires December 6, 2004

[Page 128]

Replaced Hashtable as parameter with Map in bind() and getSaslBindProperties.

The LDAP_PROPERTY_SDK is of type String rather than Float.
LDAP_PROPERTY_PROTOCOL is of type Integer.

setInputStream and setOutputStream throw LDAPException.

Removed setProperty.

Added stopTLS.

LDAPCompareAttrNames

Implements java.util.Comparator instead of LDAPEntryComparator.

LDAPConstraints

Implements Cloneable.

getServerControls renamed to getControls.

setServerControls renamed to setControls.

Added getProperty and setProperty.

getClientControls and setClientControls removed.

LDAPControl

Removed all references to "client controls".

LDAPEntryComparator

Removed; LDAPCompareAttrNames implements java.util.Comparator instead.

LDAPException

Renamed errorCodeToString() to resultCodeToString().

Renamed getLDAPResultCode () to getResultCode ().

LDAPListener

Expires December 6, 2004

[Page 129]

Renamed to LDAPMessageQueue.

LDAPMatchingRuleSchema

Combined the two constructors with explicit field parameters.

LDAPModificationSet

Removed (replaced with LDAPModification[] as parameter where referenced).

LDAPResponseListener

Renamed to LDAPResponseQueue.

LDAPRebind

Renamed to LDAPAuthHandler.

LDAPRebindAuth

Renamed to LDAPAuthProvider.

LDAPSchema

Renamed getAttribute() to getAttributeSchema(), getAttributes() to getAttributeSchemas(), getObjectClass() to getObjectClassSchema(), getSyntax() to getSyntaxSchema(), getSyntaxes() to getSyntaxSchemas(), getDITContentRule() to getDITContentRuleSchema(), getDITContentRules() to getDITContentRuleSchemas(), getDITStructureRule() to getDITStructureRuleSchema(), getDITStructureRules() to getDITStructureRuleSchemas(), getMatchingRule() to getMatchingRuleSchema(), getMatchingRules() to getMatchingRuleSchemas(), getMatchingRuleUse() to getMatchingRuleUseSchema(), getMatchingRuleUses() to getMatchingRuleUseSchemas(), getNameForm() to getNameFormSchema(), getNameForms() to getNameFormSchemas().

Added add(), modify(), remove(), and saveSchema().

LDAPSchemaElement

Renamed `getValue()` to `toString()`.

Expires December 6, 2004

[Page 130]

Changed getName() to getNames(), removed getAliases().

Removed add(), modify(), and remove().

LDAPSearchConstraints

Added constructor that takes LDAPConstraints as parameter.

LDAPSearchListener

Renamed to LDAPSearchQueue.

LDAPSearchResults

Does not implement Enumeration.

Removed nextElement().

Renamed hasMoreElements() to hasMore().

Removed sort() (sorting can now be done with classes/interfaces of the Collections framework now that LDAPAttributeSet implements Set).

LDAPSocketFactory

Renamed makeSocket() to createSocket().

LDAPUrl

Implements Cloneable.

Renamed getUrl() to toString().

Added extensions to the constructor that takes all fields explicitly.

All schema elements

Constructors take a parameter "String[] names" instead of "String name" and "String[] aliases".

Implementation Considerations

Expires December 6, 2004

[Page 131]

Removed the specification of package name for controls. The package name may be specified in a follow-on document describing various controls.

Moved the package name up to [section 1](#).

Added Dependencies.

Serializability

Made the following classes Serializable:

- LDAPAttribute
- LDAPAttributeSet
- LDAPConstraints
- LDAPControl
- LDAPEntry
- LDAPExtendedOperation
- LDAPMessage
- LDAPModification
- LDAPSchema
- LDAPSchemaElement
- LDAPSearchListener
- LDAPSearchResults
- LDAPUrl

[9.5](#) Changes from ldap-java-api-14.txt

LDAPAttributeSchema

Changed isModifiable() to isUserModifiable()

LDAPConnection

Added bind() signatures that take byte[] as password parameter, removed bind() signatures which do not take a version parameter

Use Hashtable, not Properties, in all SASL bind signatures

Removed setSearchConstraints()

rename() takes newParentdn before deleteOldRdn (one of the eight signatures had the order reversed)

LDAPException

Defined how the toString method overrides the default toString behavior

Expires December 6, 2004

[Page 132]

LDAPMatchingRuleSchema

Changed return of getSyntaxString from String[] to String

LDAPUrl

Removed constructor signature which takes "secure" as a parameter

Removed isSecure()

getFilter() returns null instead of "(objectclass=*)" if no filter was specified

Examples

Use SASL to bind in synchronous ModifyEmail example and TLS in asynchronous ModifyEmail example (instead of simple bind)

General

Put methods of LDAPAttributeSchema in alphabetical order

Clarifications and editorial changes

[9.6](#) Changes from ldap-java-api-13.txt

Notice of disconnection, Invalid responses, Level of compatibility

New section

Result codes

Added INVALID_RESPONSE, AMBIGUOUS_RESPONSE

Removed PARAM_ERROR

LDAPConnection

Removed getAuthenticationPassword()

Added bind() signatures that take a byte array for password
bind() signatures for SASL take an authzId parameter

Expires December 6, 2004

[Page 133]

`disconnect()` may take an `LDAPConstraints` parameter
`read()` throws `LDAPException` with `AMBIGUOUS_RESPONSE` if there is more than one result

`LDAPConstraints`

Removed `getReferralHandler`

`LDAPRebindAuth`

Added a signature of the constructor which takes `byte[]` as parameter
`getPassword()` returns `byte[]` instead of `String`

`LDAPReferralException`

Clarified that `getReferrals()` must return LDAP URL strings with the scope rewritten if necessary (for `SearchResultReferences` on search continuation).

`LDAPDN`

Added `normalize()` and `isValid()`.

`LDAPUnsolicitedNotificationListener`

The argument to `messageReceived()` is `LDAPExtendedResponse` and not `LDAPMessage`.

Security Considerations

Added implementation guidelines

General

All classes and methods in alphabetical order
Many clarifications and editorial changes

[9.7](#) Changes from `ldap-java-api-12.txt`

Abstract

Expires December 6, 2004

[Page 134]

Removed references to [RFC 1823](#) and to an earlier draft on an asynchronous interface.

LDAPConnection

Under clone(), added a listing of which methods affect an individual clone and which ones affect all related clones.

LDAPException

Can take Throwable as root exception argument in an additional constructor signature. Added getCause() to return the root exception. Removed reference to LDAP_PARTIAL_RESULTS among result codes.

LDAPBind

bind() throws LDAPReferralException instead of LDAPException.

LDAPReferralException

Can take Throwable as root exception argument instead of LDAPException. getFailureException() was removed and replaced by getCause() in LDAPException. Added getFailedReferral() and setFailedReferral().

LDAPControl

Removed newInstance(). An implementation of the API can instantiate a control using the LDAPControl constructor. Removed a sentence saying that LDAPv2 doesn't support controls.

Referral handling

Added section outlining the handling of exceptions on referrals.

References

Added references to RFCs 2222 and 2830.

Host names

May be IPv6 references as well as hostnames or IPv4 addresses.

Expires December 6, 2004

[Page 135]

LDAPMessage

Restored the values of the message types to correspond to the message type values in ldap-java-api-11.txt.

LDAPEntry

getAttribute returns a single attribute.

Unsolicited notifications in LDAPConnection

Removed getUnsolicitedNotifications and setUnsolicitedNotifications. Added addUnsolicitedNotificationListener and removeUnsolicitedNotificationListener. Added interface LDAPUnsolicitedNotificationListener.

[9.8](#) Changes from ldap-java-api-11.txt

LDAPConnection

Eliminated the interfaces (LDAPv2 and LDAPv3) that LDAPConnection implemented.

Eliminated setOption and getOption (there are corresponding properties in LDAPConstraints and LDAPSearchConstraints).

Removed the signatures of connect which took DN, password, and protocol version as arguments. The previous signatures were utility methods that combined connect and bind.

Added a signature of extendedOperation which takes LDAPConstraints as argument.

Added isTLS.

Added getProtocolVersion.

Added signatures of abandon which take LDAPConstraints as argument.

extendedOperation returns LDAPExtendedResponse, not LDAPExtendedOperation.

Added signatures of SASL bind that take LDAPConstraints as argument.

Expires December 6, 2004

[Page 136]

Added `getSaslBindProperties`.

Added `getSaslBindCallbackHandler`.

Added `getUnsolicitedNotifications` and `setUnsolicitedNotifications`.

The default protocol version to bind with is LDAPv3, not LDAPv2.

Constants: `LDAP_DEREF_NEVER`, etc changed to `DEREF_NEVER`, etc. Values of `DEREF_SEARCHING` and `DEREF_FINDING` corrected to those of [RFC 2251](#). The values are now defined in `LDAPSearchConstraints` instead of in `LDAPConnection`.

`LDAPControl`

Added `setValue`.

`LDAPExtendedOperation`

Added `setValue`.

`LDAPSearchResultReference`

Changed `getURLs` to `getReferrals`, which returns `String[]`.

`LDAPUrl`

Added method `isSecure`.

Added constructors that take a boolean for `isSecure`.

`LDAPReferralException`

Added `getReferralFailureException`.

Changed `getURLs` to `getReferrals`, which returns `String[]`.

`LDAPConstraints`

Takes a new interface `LDAPReferralHandler` as parameter, instead of `LDAPBind` and `LDAPRebind`.

Changed `getReferrals` to `getReferralFollowing`.

Changed setReferrals to setReferralFollowing.

Expires December 6, 2004

[Page 137]

LDAPReferralHandler

New common ancestor to LDAPBind and LDAPRebind.

LDAPListener

New common ancestor to LDAPSearchListener and LDAPResponseListener.

LDAPSearchListener

Implements LDAPListener.

Added signature of isResponseReceived and of getResponse that takes a message ID as parameter.

Added isComplete.

LDAPResponseListener

Implements LDAPListener.

Added signatures isResponseReceived and getResponse that take a message ID as parameter.

LDAPBind

Changed the signature of bind to return LDAPConnection instead of void, and take an LDAP URL string list as argument.

LDAPException

Defined the values of symbolic result codes generated by the interface. Added a constructor that takes matchedDN as parameter.

LDAPMessage

Redefined the values of the message types to correspond to the message type values in [[LDAPv3](#)].

[9.9](#) Changes from ldap-java-api-10.txt

Overview of the LDAP model

Expires December 6, 2004

[Page 138]

Allowed for character set conversion from/to T.61 (in addition to UTF-8).

LDAPConnection

Added startTLS. Added STRING_FORMAT option to setOption. Added numerical values to options in setOption and search. Changed REFERRALS_AUTHENTICATION to REFERRALS_REBIND_PROC. Clarified that getAuthenticationPassword returns null if no simple bind has been performed. Added asynchronous methods (from [\[TLS\]](#)).

LDAPConstraints

Default for setHopLimit is 10, not 5.

LDAPUrl

Added getScope and getExtensions.

Schema classes

Added parameters to constructors (aliases, obsolete, collective).

Various classes

Removed all "synchronized" qualifier on methods. Added a statement that implementations should ensure thread-safety.

[9.10](#) Changes from ldap-java-api-09.txt

Overview of LDAP API use

Clarifications were added on the behavior of the SDK for null values of LDAPConstraints and for a DN.

LDAPAttributeSet

Return type of getAttribute is LDAPAttribute, not LDAPAttribute[].

LDAPV2

Expires December 6, 2004

[Page 139]

Added convenience method of add that does not take LDAPConstraints, added read method that does take LDAPSearchConstraints.

Error Codes

Changed to Result Codes. Added TLS_NOT_SUPPORTED.

LDAPSearchResults

Clarified in declaration that it implements Enumeration.

LDAPV3

Added constants NO_ATTRS and ALL_USER_ATTRS.

Schema classes

Added LDAPDITContentRuleSchema, LDAPDITStructureRuleSchema, LDAPMatchingRuleUseSchema, LDAPNameFormSchema, LDAPSyntaxSchema.

[9.11](#) Changes from ldap-java-api-08.txt

Standards track

Added intended category to first page header.

SASL references

Removed all references to a Java SASL internet draft.

LDAPv2

Removed static methods search(LDAPUrl url). The methods are still present in LDAPConnection.

[9.12](#) Changes from ldap-java-api-07.txt

LDAPAttributeSchema

Removed getAliases() because it is already defined in the superior class LDAPSchemaElement. Removed getSyntax() which

returned an integer.

Expires December 6, 2004

[Page 140]

LDAPConnection

Added `getAuthenticationMethod()`.

LDAPSchemaElement

Changed `getOID()` to `getID()`.

9.13 Changes from `ldap-java-api-06.txt`

LDAPAttributeSchema

Added a constructor that takes the attribute syntax as a String, an optional superior attribute type, and an optional list of aliases. Removed previous constructor.
Added `getSuperior()` and `getSyntaxString()`.

LDAPConnection

Added `getInputStream()`, `getOutputStream()`, `setInputStream()` (Error! Reference source not found.), and `setOutputStream()`. They are used when establishing a security layer with SASL, and may also be used to interpose a proxy.

LDAPDN

Added `equals()`.

LDAPException

Added additional error codes defined in [\[URL\]](#).

LDAPMatchingRuleSchema

Added a constructor that takes the attribute syntax as a String and an optional list of aliases. Removed previous constructor.

LDAPObjectClassSchema

Added a constructor that takes an array of superior object class names, a type (ABSTRACT, AUXILIARY, or STRUCTURAL), and an optional list of aliases. Removed previous constructor.
Added `getSuperiors()` and `getType()`. Removed `getSuperior()`.

Expires December 6, 2004

[Page 141]

LDAPSchemaElement

Added overloaded methods of add, remove, and modify which take a DN as parameter, for specifying where in the DIT to determine the subschemaSubentry for the modification.
Added getAliases(), getQualifier(), getQualifierNames(), isObsolete(), and setQualifier().

[9.14](#) Changes from ldap-java-api-05.txt

LDAPConnection

Distinguished between getConstraints() and getSearchConstraints(), and between setConstraints() and setSearchConstraints().

LDAPConstraints

LDAPBind and LDAPRebind should not be specified in the same constructor. Added setClientControls().

LDAPSearchConstraints

LDAPBind and LDAPRebind should not be specified in the same constructor.

LDAPControl

newInstance() is now static.

LDAPv3

Changed the signature of the bind() methods to match the Java SASL Internet Draft.

[9.15](#) Changes from ldap-java-api-04.txt

LDAPAttribute

Added getByteArrayArray() and getStringValueArray().

LDAPCompareAttrNames

Expires December 6, 2004

[Page 142]

Added `getLocale()` and `setLocale()`.

LDAPSchemaElement

Added `modify()`.

LDAPSchemaElement

Added `fetchSchema(LDAPConnection, String)`.

[9.16](#) Changes from `ldap-java-api-03.txt`

LDAPBind

New interface, to support sophisticated reauthentication mechanisms.

LDAPControl

Added methods `register()` and `newInstance()`, to support dynamic registration and instantiation of server response controls.

LDAPConstraints

Separated interface time limit from server search time limit.
Moved all search-only constraints to `LDAPSearchConstraints`.

LDAPRebind

Reverted back to original name, instead of `LDAPReauthentication` as it was in the previous draft.

LDAPRebindProc

Reverted back from `LDAPCredentials`.

LDAPSearchConstraints

Reinstated this class, to represent all constraints applicable to a search. `LDAPConstraints` (which it extends) only represents common constraints for all operations.

LDAPSearchResults

Expires December 6, 2004

[Page 143]

Added `getResponseControls()`.

LDAPv2

Added `abandon()`. Separated interface time limit from server search time limit. Changed `authenticate()` to `bind()`.

LDAPv3

Changed `authenticate()` to `bind()`.

[9.17](#) Changes from `ldap-java-api-02.txt`

LDAPSearchConstraints

Renamed to `LDAPConstraints`, since it can be applied to operations other than search.

LDAPRebind

Renamed to `LDAPReauthentication`. Added a definition of its single method.

LDAPRebindProc

Renamed to `LDAPCredentials`.

[9.18](#) Changes from `ldap-java-api-01.txt`

LDAPAttribute

Added a copy constructor.
Added support for subtypes, and for language subtypes in particular.

LDAPAttributeSet

`LDAPAttributeSet` implements `Cloneable`.
Added `getSubset()` for subtype support.

LDAPDN

Expires December 6, 2004

[Page 144]

Added support for escaping and unescaping an RDN.

LDAPException

Added the SASL_BIND_IN_PROGRESS error code.

LDAPSearchResults

Added getCount(), to report the number of results returned.

LDAPConnection

Added a signature that passes LDAPConstraints to read(LDAPURL).

LDAPv2

Added signatures that pass LDAPConstraints to the following methods:

- add()
- compare()
- modify()
- read()
- rename()

LDAPv3

Removed "Preferred Language", because it has been dropped from the extension work.

Added a signature that passes LDAPConstraints to rename().

Expires December 6, 2004

[Page 145]