Abstract

This document defines asynchronous extensions to the java language application program interface to the lightweight directory access protocol (LDAP) defined in [JAVALDAP].
1. Introduction.....................................................4
2. Overview of the LDAP asynchronous extension classes........4
2.1 Interfaces....................................................4
2.2 Classes........................................................4
3. Overview of extended asynchronous LDAP API use.............5
4. The java LDAP asynchronous extension classes..............5
4.1 public interface LDAPAsynchronousConnection.............5
4.1.1 abandon...................................................6
4.1.2 add.........................................................6
4.1.3 bind.........................................................6
4.1.4 compare....................................................7
4.1.5 delete.......................................................8
4.1.6 modify......................................................8
4.1.7 rename......................................................9
4.1.8 search......................................................10
4.2 public class LDAPExtendedResponse extends LDAPResponse 11
4.2.1 getOID......................................................11
4.2.2 getValue...................................................11
4.3 public class LDAPMessage...................................11
4.3.1 getControls................................................11
4.3.2 getID.......................................................11
4.3.3 getType....................................................12
4.4 public abstract class LDAPResponse extends LDAPMessage 12
4.4.1 getErrorMessage..........................................12
4.4.2 getMatchedDN...............................................12
4.4.3 getReferrals...............................................12
4.4.4 getResultCode............................................13
4.5 public class LDAPResponseListener..........................13
4.5.1 getIDs......................................................13
4.5.2 getResponse................................................13
4.5.3 isResponseReceived......................................13
4.5.4 merge......................................................13
4.6 public class LDAPSearchListener............................13
4.6.1 getIDs......................................................13
4.6.2 getResponse................................................14
4.6.3 isResponseReceived......................................14
4.6.4 merge......................................................14
4.7 public class LDAPSearchResult extends LDAPMessage......14
4.7.1 getEntry....................................................14
4.8 public class LDAPSearchResultReference extends LDAPMessage.14
4.8.1 getUrls.....................................................14
5. Bibliography....................................................14
6. Authors’ Addresses.............................................16
7. Appendix A - Sample java LDAP asynchronous extension programs...17
8. Appendix B - Changes from draft-ietf-ldapext-ldap-java-api-asynch-ext-01.txt..................................................20
8.1 LDAPResponseListener........................................20
9. Appendix C - Changes from draft-ietf-ldapext-ldap-java-api-asynch-ext-00.txt..................................................21
9.1 LDAPAsynchronousConnection................................21
9.2 LDAPMessage..................................................21
1. Introduction

The LDAP class library defined in [JAVALDAP] is designed to provide powerful, yet simple, access to LDAP directory services. It defines a synchronous interface to LDAP, with support for partial results on searching, to suit a wide variety of applications. However, in some cases it is advantageous for a programmer to access the underlying asynchronous mechanisms close to the protocol layer. This document describes an extension interface - LDAPAsynchronousConnection - and supporting classes. The public class methods are described in detail, followed by an appendix that provides some example code demonstrating the use of the classes.

2. Overview of the LDAP asynchronous extension classes

The central element is the interface LDAPAsynchronousConnection. It provides methods to authenticate to an LDAP server, as well as methods to search for, modify, compare, and delete entries in the directory.

Unlike the LDAPv3 interface defined in [JAVALDAP], LDAPAsynchronousConnection returns a listener object and may also take a listener object as input. The listener is a message queue associated with the request, and it is the responsibility of the client to read messages out of the queue and process them.

Messages retrieved from an LDAPResponseListener are result objects derived from LDAPResponse. Messages retrieved from an LDAPSearchListener are either result objects derived from LDAPResponse, search results, or search result references.

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

2.1 Interfaces

LDAPAsynchronousConnection Encapsulates a connection to an LDAP server, providing access to the input queue for messages received.

2.2 Classes

LDAPMessage Base class for LDAP request and response messages.
LDAPResponse  Class representing a message received from an LDAP server in response to a request. It extends LDAPMessage.

LDAPExtendedResponse  The response returned by an LDAP server on an extended operation request. It extends LDAPResponse.

LDAPSearchResult  A single search result. It extends LDAPMessage.

LDAPSearchResultReference  A continuation reference from a search operation. It extends LDAPMessage.

LDAPResponseListener  Low-level mechanism for processing messages received from a server.

LDAPSearchListener  Low-level mechanism for queueing search results received from a server.

3. Overview of extended asynchronous LDAP API use

An application generally uses the asynchronous methods as follows:

- Construct an LDAPAsynchronousConnection. Initialize an LDAP session with a directory server.

- Invoke an LDAP operation, passing null for LDAPResponseListener (or LDAPSearchListener for a search operation). The LDAPAsynchronousConnection returns a new LDAPResponseListener (or LDAPSearchListener for a search operation).

- Loop on reading from the listener object, which blocks until there is a response available, until the operation has completed, and interpret the results.

An LDAPResponseListener 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.

The following sections describe the asynchronous extension classes in more detail.

4. The java LDAP asynchronous extension classes

4.1 public interface LDAPAsynchronousConnection
LDAPAsynchronousConnection provides access to the message queue in a connection to a server, allowing clients to multiplex results from multiple connections or do low-level processing of incoming messages.

### 4.1.1 abandon

```java
public void abandon(int id)
public void abandon(LDAPsearchListener listener)
```

Abandons one or all search operations for a listener.

Parameters are:

- **id**
  - The ID of the operation to abandon. The ID may be obtained from the search listener for the operation.

- **listener**
  - Handler returned for messages returned on a search request. All operations in progress which are managed by the listener are abandoned.

### 4.1.2 add

```java
public LDAPResponseListener add(LDAPEntry entry, LDAPResponseListener listener)
throws LDAPException
public LDAPResponseListener add(LDAPEntry entry, LDAPResponseListener listener, LDAPConstraints cons)
throws LDAPException
```

Adds an entry to the directory.

Parameters are:

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

- **listener**
  - Handler for messages returned from a server in response to this request. If it is null, a listener object is created internally.

- **cons**
  - Constraints specific to the operation.

### 4.1.3 bind

```java
public LDAPResponseListener bind(String dn,
```

Expires April 8, 2000
public LDAPResponseListener bind(String dn,
String passwd,
LDAPResponseListener listener)
throws LDAPException

Authenticates to the LDAP server (that the object is currently connected to) using the specified name and password. If the object has been disconnected from an LDAP server, this method attempts to reconnect to the server. If the object had already authenticated, the old authentication is discarded.

Parameters are:

- **dn** If non-null and non-empty, specifies that the connection and all operations through it should be authenticated with dn as the distinguished name.
- **passwd** If non-null and non-empty, specifies that the connection and all operations through it should be authenticated with dn as the distinguished name and passwd as password.
- **listener** Handler for messages returned from a server in response to this request. If it is null, a listener object is created internally.
- **cons** Constraints specific to the operation.

### 4.1.4 compare

```java
public LDAPResponseListener compare(String dn,
LDAPAttribute attr,
LDAPResponseListener listener)
throws LDAPException

public LDAPResponseListener compare(String dn,
LDAPAttribute attr,
LDAPResponseListener listener,
LDAPConstraints cons)
throws LDAPException
```

Compare an attribute value with one in the directory.

Parameters are:
dn             The distinguished name of the entry containing an
             attribute to compare.
attr           An attribute to compare.
listener       Handler for messages returned from a server in
               response to this request. If it is null, a
               listener object is created internally.
cons           Constraints specific to the operation.

4.1.5 delete

public LDAPResponseListener delete(String dn
                                       LDAPResponseListener listener)
       throws LDAPException

public LDAPResponseListener delete(String dn,
                                       LDAPResponseListener listener,
                                       LDAPConstraints cons)
       throws LDAPException

Deletes the entry for the specified DN from the directory.

Parameters are:

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

4.1.6 modify

public LDAPResponseListener modify(String dn,
                                       LDAPModification mod,
                                       LDAPResponseListener listener)
       throws LDAPException

public LDAPResponseListener modify(String dn,
                                       LDAPModification mod,
                                       LDAPResponseListener listener,
                                       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.

```java
public LDAPResponseListener modify(String dn,
        LDAPModificationSet mods,
        LDAPResponseListener listener)
        throws LDAPException
```

```java
public LDAPResponseListener modify(String dn,
        LDAPModificationSet mods,
        LDAPResponseListener listener,
        LDAPConstraints cons)
        throws LDAPException
```

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

Parameters are:

- **dn** Distinguished name of the entry to modify.
- **mod** A single change to be made to the entry.
- **mods** A set of changes to be made to the entry.
- **listener** Handler for messages returned from a server in response to this request. If it is null, a listener object is created internally.
- **cons** Constraints specific to the operation.

### 4.1.7 rename

```java
public LDAPResponseListener rename(String dn,
        String newRdn,
        boolean deleteOldRdn,
        LDAPResponseListener listener)
        throws LDAPException
```

```java
public LDAPResponseListener rename(String dn,
        String newRdn,
        boolean deleteOldRdn,
        LDAPResponseListener listener,
        LDAPConstraints cons)
        throws LDAPException
```

Renames an existing entry in the directory.
Parameters are:

dn             Current distinguished name of the entry.
newRdn         New relative distinguished name for the entry.
deleteOldRdn   If true, the old name is not retained as an attribute value.
listener       Handler for messages returned from a server in response to this request. If it is null, a listener object is created internally.
cons           Constraints specific to the operation.

4.1.8 search

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

Performs the search specified by the parameters.

public LDAPSearchListener search(String base,
int scope,
String filter,
String attrs[],
boolean typesOnly,
LDAPSearchListener listener,
LDAPConstraints cons)
throws LDAPException

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

Parameters are:

base           The base distinguished name to search from.
scope          The scope of the entries to search. The following are the valid options:
LDAPv2.SCOPE_ONE  Search only entries under the base DN

LDAPv2.SCOPE_SUB  Search the base DN and all entries within its subtree

filter         Search filter specifying the search criteria, as defined in [3].

attrs          Names of attributes to retrieve.

typesOnly      If true, returns the names but not the values of the attributes found. If false, returns the names and values for attributes found.

listener       Handler for messages returned from a server in response to this request. If it is null, a listener object is created internally.

cons           Constraints specific to the search.

4.2 public class LDAPExtendedResponse extends LDAPResponse

An LDAPExtendedResponse object encapsulates a server response to an extended operation request.

4.2.1 getOID

public String getOID()

Returns the OID of the response.

4.2.2 getValue

public byte[] getValue()

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

4.3 public class LDAPMessage

Base class for LDAP request and response messages.

4.3.1 getControls

public LDAPControl[] getControls()

Returns any controls in the message.

4.3.2 getID
public int getID()

Returns the message ID.

4.3.3 getType

public int getType()

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

```java
public final static int BIND_REQUEST = 0;
public final static int BIND_RESPONSE = 1;
public final static int UNBIND_REQUEST = 2;
public final static int SEARCH_REQUEST = 3;
public final static int SEARCH_RESPONSE = 4;
public final static int SEARCH_RESULT = 5;
public final static int MODIFY_REQUEST = 6;
public final static int MODIFY_RESPONSE = 7;
public final static int ADD_REQUEST = 8;
public final static int ADD_RESPONSE = 9;
public final static int DEL_REQUEST = 10;
public final static int DEL_RESPONSE = 11;
public final static int MODIFY_RDN_REQUEST = 12;
public final static int MODIFY_RDN_RESPONSE = 13;
public final static int COMPARE_REQUEST = 14;
public final static int COMPARE_RESPONSE = 15;
public final static int ABANDON_REQUEST = 16;
public final static int SEARCH_RESULT_REFERENCE = 19;
public final static int EXTENDED_REQUEST = 23;
public final static int EXTENDED_RESPONSE = 24;
```

4.4 public abstract class LDAPResponse extends LDAPMessage

Represents the response to a particular LDAP operation.

4.4.1 getErrorMessage

public String getErrorMessage()

Returns any error message in the response.

4.4.2 getMatchedDN

public String getMatchedDN()

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

4.4.3 getReferrals

public String[] getReferrals()
Returns all referrals, if any, in a server response.

4.4.4 getResultCode

public int getResultCode()

Returns the result code in a server response.

4.5 public class LDAPResponseListener

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

4.5.1 getIDs

public int[] getIDs()

Returns the message IDs for all outstanding requests.

4.5.2 getResponse

public LDAPResponse getResponse()

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

4.5.3 isResponseReceived

public boolean isResponseReceived()

Reports true if a response has been received from the server.

4.5.4 merge

public void merge(LDAPResponseListener listener2)

Merges two response listeners. Moves/appends the content from another listener to this one.

4.6 public class LDAPSearchListener

An LDAPSearchListener manages search results and references returned on one or more search requests.

4.6.1 getIDs

public int[] getIDs()

Returns the message IDs for all outstanding requests.
4.6.2 getResponse

    public LDAPMessage getResponse()

Blocks until a response is available, or until all operations associated with the object have completed or been canceled, and returns the response. The response may be a search result, a search reference, a search response, or null (if there are no more outstanding requests). LDAPException is thrown on network errors.

4.6.3 isResponseReceived

    public boolean isResponseReceived()

Reports true if a response has been received from the server.

4.6.4 merge

    public void merge(LDAPSearchListener listener2)

Merges two search listeners. Moves/appends the content from another search listener to this one.

4.7 public class LDAPSearchResult extends LDAPMessage

    An LDAPSearchResult object encapsulates a single search result.

4.7.1 getEntry

    public LDAPEntry getEntry()

Returns the entry of a server search response.

4.8 public class LDAPSearchResultReference extends LDAPMessage

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

4.8.1 getUrls

    public String[] getUrls()

Returns any URLs in the object.

5. Bibliography


Expires April 8, 2000
6. Authors’ Addresses

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

Christine Tomlinson
Innosoft International, Inc.
8911 Capital of Texas Highway
Suite 4140
Austin, TX US 78759
+1 512 231 1600
christine.tomlinson@innosoft.com

Miodrag Kekic
Netscape Communications Corp.
501 E. Middlefield Rd.
Mountain View, CA 94043
USA
+1 650 937-5663
miodrag@netscape.com

Expires April 8, 2000
import netscape.ldap.*;
import java.util.*;

/* Assume a class LDAPAsynch which extends LDAPConnection and
   implements LDAPAsynchronousConnection */

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

            /* Asynchronous authentication */
            LDAPResponseListener r =
                ld.bind( "uid=admin, o=Airius.com",
                "password", null );

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

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

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

            LDAPSearchListener l =
                ld.search( MY_SEARCHBASE,
                        ld.SCOPE_ONE,
                        MY_FILTER,
                        null,
                        false,
                        null,
                        (LDAPSearchListener)null );

            /* Loop on results until finished */
            LDAPMessage msg;
            while( (msg = l.getResponse()) != null ) {
                if ( msg instanceof LDAPSearchResultReference ) {
                    /* Do something */
                }
            }
        }
    }
}
String[] urls =
    ((LDAPSearchResultReference)msg).getUrls();
// 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.errorCodeToString(status);
        throw new LDAPException(
            err,
            status,
            res.getErrorMessage(),
            res.getMatchedDN() );
    }
}
} catch ( LDAPException e ) {
    System.err.println( e.toString() );
}
/* Done, so disconnect */
if ( ld.isConnected() ) {
    ld.disconnect();
}
}
import netscape.ldap.*;
import java.util.*;

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

public class MultiplexServers {
    public static void main( String[] args )
    {
        try {
            LDAPAsynch[] ld = new LDAPAsynch[3];
            String[] hosts = { "foo1", "foo2", "foo3" };
            int[] ports = { 389, 389, 2018 }
            String[] bases =
            { "o=Airius.com", "o=Acme.com", "dc=Acme,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 LDAPAsynch();
                /* Connect to server */
                ld[i].connect( hosts[i], ports[i] );
            }

            /* Get a response listener for one search */
            LDAPSearchListener l =
            ld[0].search( bases[0],
            ld.SCOPESUB,
            MY_FILTER,
            null,
            false,
            (LDAPSearchListener)null );

            /* Share the listener */
            for( i = 1; i < ld.length; i++ ) {
                ld[i].search( bases[i],
                ld[i].SCOPESUB,
                MY_FILTER,
                null,
                false,
                1 );
            }

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

Expires April 8, 2000
import netscape.ldap.*;
import java.util.*;

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

class MultiplexTrees {
    public static void main( String[] args )
    {
        try {
            LDAPAsynch ld = new LDAPAsynch();
            /* Connect to server */
            String MY_HOST = "localhost";
            int MY_PORT = 389;
            ld.connect( MY_HOST, MY_PORT );
            String MY_FILTER = "sn=Jensen";
            String[] bases =
                { "o=Airius.com", "o=Acme.com", "dc=Acme,dc=com" };

            /* Get a response listener for one search */
            LDAPSearchListener l =
                ld.search( bases[0],
                   ld.SCOPE_SUB,
                   MY_FILTER,
                   null,
                   false,
                   (LDAPSearchListener)null );

            /* Share the listener */
            for( 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 */
            /* ... */
        }
    }

    8. Appendix B - Changes from draft-ietf-ldapext-ldap-java-api-asynch-
    ext-01.txt

    8.1 LDAPResponseListener

    Added merge(), which was already present in LDAPSearchListener in the
    previous draft.
Appendix C - Changes from draft-ietf-ldapext-ldap-java-api-asynch-ext-00.txt

9.1 LDAPAsynchronousConnection
   Added abandon(), compare(), and unbind(). Removed bind() method for SASL. Removed static search methods.

9.2 LDAPMessage
   No longer abstract. Added getID().

9.3 LDAPResponse
   No longer abstract.

9.4 LDAPResponseListener
   Added getIDs().

9.5 LDAPSearchListener
   No longer extends LDAPResponseListener. Removed getSearchResult(), added getIDs(), getResponse(), isResponseReceived(), and merge().

9.6 Appendix
   Updated sample programs to reflect API changes.