

January 1, 1998

[Tutorial Index](#)

by Alan Frank

## Lesson 113: Lightweight Directory Access Protocol

An offshoot of X.500, LDAP may be the simple, universal directory of the future.

This lesson relates closely to "[Lesson 109: Directories](#)" (*Network Magazine*, September 1997, page 25, by Alan Frank), which discusses directories in general. This month I will focus on a specific directory technology: the Lightweight Directory Access Protocol (LDAP).

As "LDAP" suggests, it started life as an access method, not a directory. By establishing a protocol, LDAP standardizes how an application can "talk" to a directory. LDAP was originally designed to be a lightweight front end to X.500 directories. X.500 is the ISO's robust and comprehensive directory standard.

However, at least in its early stages, X.500 took quite a bit of computing power to implement. Researchers at the University of Michigan conceived LDAP as a simplified replacement for X.500's native access protocol, known as the Directory Access Protocol. An X.500 directory could be run on a mainframe, minicomputer, or other high-horsepower machine, while individuals with PCs and other computationally challenged machines could use LDAP to access the directory.

Despite LDAP's beginnings as a front end to X.500 directories, work at the University of Michigan soon turned to developing a standalone LDAP-based directory that could be accessed by any LDAP client. Since then, IETF standards efforts have also been directed toward standalone LDAP directories.

### THE LDAP STANDARD(S)

Efforts to make LDAP an Internet standard resulted in RFC 1487, which describes LDAP 1. This document has since been superseded by RFC 1777, which covers LDAP 2. The IETF working group responsible for Access, Searching, and Indexing of Directories (ASID) has since developed a proposal for LDAP 3 and submitted it for approval in the fall of 1997. For all practical purposes, it has been approved, but an

RFC has yet to be issued (at least at the time of this writing). The draft document for LDAP 3 can be found at <ftp://ftp.ietf.org/internet-drafts/draft-ietf-asid-ldapv3-protocol-08.txt>.

In theory, any LDAP-compliant application should be able to access any LDAP-compliant directory. The LDAP-compliant directory would then play the role of a server, while the application would function as an LDAP client. LDAP can be used to get information from a directory (such as finding out a user's e-mail address), as well as to store information (such as creating a new account).

## **HOW IT WORKS**

The goal of a directory is to keep track of some entity, which might be a person, but could also be a printer or other resource, and to store relevant and needed information about that entity. An LDAP directory is organized into entries; there is an entry for each entity you want to keep track of. Each entry has attributes (for example, a name or e-mail address) that hold information about the entity. Each attribute has a name, a type, and one or more associated values. Take, for example, the network user John Doe. The entry for John in the directory may have an attribute named "mail," which stores John's e-mail address. The attribute type of the mail attribute might be String, which means that the information is stored as an ASCII text string, and the attribute value of the mail attribute for John Doe's entry might be `jdoe@xyz.com`.

Since you need to keep track of different attributes for a person than for a printer, entries have a type associated with them. An entry's type is known as its object class. An entry's object class defines what attributes it will have. How do we know what an entry's type is? That's held in a special attribute named `objectClass`.

What attributes are stored for a given object type is known as an object's schema. The schema is enforced according to an entry's object class, but one of the major extensions contained in LDAP 3 is the provision for an extensible schema. The LDAP 3 proposal defines a special object class called `extensibleObject`. If an entry's `objectClass` attribute contains the value `extensibleObject`, anything goes as far as what attributes are allowed. In other words, schema enforcement is overruled.

LDAP 3 also provides a method for clients to discover the schema, which is a very important, much-needed capability. For example, is a Social Security number or address information in the directory? If so, what's the attribute name? Schema discovery gives the LDAP client a means to find this out. In LDAP 1 and LDAP 2, a client had to know what the schema was—a major limitation.

## **NAVIGATING THE DIRECTORY**

How do you find a particular entry in an LDAP directory? LDAP supports a hierarchical organization of the information in the directory. It's based on concepts borrowed from X.500, so if you've worked with X.500 or Novell Directory Services, which bases its naming scheme on X.500, you're probably familiar with LDAP's naming convention. LDAP entries are named according to their Distinguished Name, known as DN. For example, the DN for the John Doe entry might be "cn=John Doe, o=XYZ, c=US." This specifies the entry as having the common name "John Doe" in the U.S.-based organization "XYZ."

The complete description of LDAP's naming scheme can be found in RFC 1779.

## **LDAP OPERATIONS**

An LDAP client uses TCP/IP to communicate with an LDAP server. By default, LDAP servers listen to port 389, so LDAP clients need to direct their requests to that port.

Communications between LDAP clients and servers are session-oriented, and the first thing a client must do when initiating a session is send a bind request. This establishes the session.

As part of the session setup procedure, LDAP directories require authentication of the client. LDAP 1 and LDAP 2 provide for a simple password-based authentication, but the password is sent in plain text. This is one of the issues spurring LDAP 3 work. (LDAP 1 and LDAP 2 can also use Kerberos authentication.) The LDAP 3 proposal allows for the use of something called Transport Layer Security (TLS), which is based on Secure Sockets Layer (SSL) 3.

LDAP defines several operations that an LDAP server can perform for a client. These include bind (which was already mentioned), unbind (which is issued to terminate the session), search (search for an entry), modify (modify an entry), add (add an entry for a new entity), del (delete an existing entry), modifyRDN (modify the RDN of an entry), compare (which I will explain), and abandonRequest (abandon a request that's pending).

Searches can, and typically are, limited in scope and by search criteria (known as search filters). A search filter is a means of determining what you're searching for. For example, you may be looking for any entries where "cn=Elvis." Another alternative is to do a substring search. A substring search for common names beginning with "e" might return entries for Edward Jones, Edna Edwards, and Eloise Smith, for example.

The ability to set filters is a very powerful aid in searching the directory,

and LDAP's filtering possibilities are very flexible. You can require either exact or approximate matches for your filter criteria, as well as use greater-than-or-equal-to or less-than-or-equal-to comparisons. You can also use the And, Or, and Not functions to look for matches for various combinations of attributes and values.

When submitting a search request, the client must provide a parameter called `baseObject`, which is the DN of the entry, from which point the search will begin. For example, if you're looking for someone in company XYZ, you might specify "o=XYZ, c=US" as the `baseObject`.

With `baseObject` as the starting point, how far should the search extend? That's defined by the scope parameter. Scope can have one of three values. The first is a subtree search, which means everything "below" the `baseObject` will be searched (in other words, `baseObject` will be the root of the search operation).

The second possibility for scope is a one-level search. This will search all entries for which the specified `baseObject` is the immediate parent. For example, if there was an entry for the accounting department in company XYZ, you could search for all the entries that are immediately subordinate to the accounting department entry by setting the `base-Object` to "ou=Accounting, o=XYZ, c=US," and setting the scope to one-level. ("Ou" stands for the object type Organizational Unit.) Note that if you wanted to find every single entry that falls under the organizational unit "Accounting," you would need to use the subtree scope, not one-level scope.

The third option for scope is a base-object scope, which will only return the entry for the base object itself. This is handy when you want to look up a certain attribute (or collection of attributes) for a specific entry, and when you know exactly which entry you want. For example, if you just wanted to find out John Doe's e-mail address, you might specify a `baseObject` "cn=John Doe, o=XYZ, c=US," and set the scope to a base-object search.

When an LDAP server gets a request from a client, it will execute that request, if it can, and return a response message. The response might be an error code (in cases where the request could not be honored) or the results of the operation requested by the client. In the case of a search request, for example, the LDAP server will return all entries that meet the submitted scope and filter criteria.

When submitting a search request, you also specify how much information should be returned. For example, you might specify that only the first 30 entries that are found should be returned. You can also specify how many seconds to spend on the search.

Whenever the server finds an entry that matches your search criteria, it

will return the entire entry (subject to whatever access permissions you have). But you can elect to have only certain attributes returned, such as a telephone number. Your request message must include a list of all the attributes you wish to have returned; if no attributes are specified, that is, if it's an empty list, you will get the entire entry.

The compare operation lets you check to see if the value of an attribute for a specific entry matches some value that you're testing for. You could, for example, verify that John Doe's email address is indeed `jdoe@xyz.com` by issuing a compare request for the entry "cn=John Doe, o=XYZ, c=US," and specifying the attribute/value assertion "mail=jdoe@xyz.com." The LDAP server will look up the requested entry and return a response code of `compareTrue` or `compareFalse`. If it can't execute the operation, it will return an error code.

## REFERRALS

In LDAP 1 and LDAP 2, no provision was made for LDAP servers returning referrals to clients. However, in the case of failed queries (where, for example, a search is made for an entry that does not exist in one LDAP directory, but might exist in another), LDAP 3 provides a method for the LDAP server that's getting the initial request to return a referral to another LDAP server. The LDAP client can then issue the request to the server identified in the referral.

## FOR MORE INFO

A good overall description of LDAP and LDAP concepts can be found in the book *LDAP: Programming Directory-Enabled Applications with Lightweight Directory Access Protocol*, by Tim Howes and Mark Smith (1997, Macmillan Technical Publishing). Other good sources of information are *An LDAP Roadmap & FAQ*, by Jeff Hodges of Stanford University ([www.leland.stanford.edu/group/networking/directory/x500ldapfaq.html](http://www.leland.stanford.edu/group/networking/directory/x500ldapfaq.html)), and LDAP information pages maintained by Critical Angle, a developer of LDAP servers and related technology ([critical-angle.com/ldapworld/ldapv3.html](http://critical-angle.com/ldapworld/ldapv3.html)).

## ENABLING DIRECTORY ACCESS

LDAP sets forth a standard for how network applications can access directories. It's also simple and easy enough to implement that it's been enthusiastically embraced by many vendors, so it's rapidly becoming the lingua franca of directory operations. In a world where each application and operating system has its own proprietary directory, LDAP holds forth the promise of directory interoperability. But for now, that interoperability extends for the most part only to simple browsing operations.

