CITI Technical Report 95–8

The Lightweight Directory Access Protocol: X.500 Lite

Timothy A. Howes
`tim@umich.edu`

ABSTRACT

This paper describes the Lightweight Directory Access Protocol (LDAP), which provides low-overhead access to the X.500 directory. LDAP includes a subset of full X.500 functionality. It runs directly over TCP and uses a simplified data representation for many protocol elements. These simplifications make LDAP clients smaller, faster, and easier to implement than full X.500 clients. Our freely available implementation of the protocol is also described. It includes an LDAP server and a client library that makes writing LDAP programs much easier.

July 27, 1995

Center for Information Technology Integration
University of Michigan
519 West William Street
Ann Arbor, MI 48103-4943

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

Find authenticated court documents without watermarks at docketalarm.com.

The Lightweight Directory Access Protocol: X.500 Lite

Timothy A. Howes

July 27, 1995

1. Introduction

X.500, the OSI directory standard [1], defines a comprehensive directory service, including an information model, a namespace, a functional model, and an authentication framework. X.500 also defines the Directory Access Protocol (DAP) used by clients to access the directory. DAP is a full OSI protocol that contains extensive functionality, much of which is not used by most applications.

DAP is significantly more complicated than the more prevalent TCP/IP stack implementations and requires more code and computing horsepower to run. The size and complexity of DAP make it difficult to run on smaller machines such as the PC and Macintosh where TCP/IP functionality often comes bundled with the machine. When the DAP stack implementations are used, they typically require an involved customization process, which has limited the acceptance of X.500.

The Lightweight Directory Access Protocol (LDAP) was designed to remove some of the burden of X.500 access from directory clients, making the directory available to a wider variety of machines and applications. Building on similar ideas in the DAS [7] and DIXIE [4] protocols, LDAP runs directly over TCP/IP or other reliable transport. As we shall see, it simplifies many X.500 operations, leaving out little-used features and emulating some operations with others. LDAP uses simple string encodings for most attributes. The result is a low-overhead access method for the X.500 directory, suitable for use on virtually any platform.

Section 2 of this paper gives a quick introduction to X.500. Section 3 gives an overview of LDAP, describing the simplifications it makes to X.500. Section 4 summarizes the key advantages of the LDAP protocol. Section 5 briefly describes our implementation of LDAP, including our server and client library. Section 6 compares the performance of DAP and LDAP. Finally, Section 7 describes some work we are doing that builds on LDAP.

2. Overview of X.500

X.500 is the OSI directory service. X.500 defines the following components:

- An information model—determines the form and character of information in the directory.
- A namespace—allows the information to be referenced and organized.
- A functional model—determines what operations can be performed on the information.
- An authentication framework—allows information in the directory to be secured.

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

- A distributed operation model—determines how data is distributed and how operations are carried out.

The information model is centered around *entries*, which are composed of *attributes*. Each attribute has a *type* and one or more *values*. The type determines the attribute's *syntax*, which defines what kind of information is allowed in the values.

Which attributes are required and allowed in an entry are controlled by a special *objectClass* attribute in every entry. The values of this attribute identify the type of entry (e.g., person, organization, etc.). The type of entry determines which attributes are required, and which are optional. For example, the object class *person* requires the *surname* and *commonName* attributes, but *description*, *seeAlso*, and others are optional.

Entries are arranged in a tree structure and divided among servers in a geographical and organizational distribution. Entries are named according to their position in this hierarchy by a distinguished name (DN). Each component of the DN is called a relative distinguished name (RDN). *Alias* entries, which point to other entries, are allowed, circumventing the hierarchy. Figure 1 depicts the relationship between entries, attributes, and values and shows how entries are arranged into a tree.

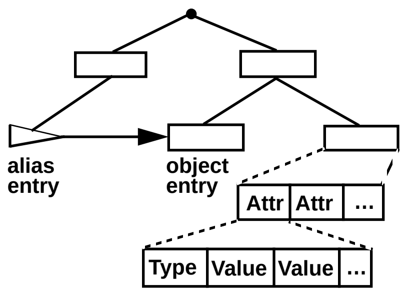

Figure 1. X.500 information model. The X.500 model is centered around entries composed of attributes that have a type and one or more values. Entries are organized in a tree structure. Alias entries can be used to build non-hierarchical relationships.

Functionally, X.500 defines operations in three areas: search and read, modify, and authenticate. In the first category, the *read* operation retrieves the attributes of an entry whose name is known. The *list* operation enumerates the children of a given entry. The *search* operation selects entries from a defined area of the tree based on some selection criteria known as a search filter. For each matching entry, a requested set of attributes (with or without values) is returned. The searched entries can span a single entry, an entry's children, or an entire subtree. Alias entries can be followed automatically during a search, even if they cross server boundaries.

In the second category, X.500 defines four operations for modifying the directory. The *modify* operation is used to change existing entries. It allows attributes and values to be added and deleted. The *add* and *delete* operations are used to insert and remove entries from the directory. The *modify RDN* operation is used to change the name of an entry.

The final category defines a *bind* operation, allowing a client to initiate a session and prove its identity to the directory. Several authentication methods are supported, from simple clear-text password to public key-based authentication. The *unbind* operation is used to terminate a directory session. An *abandon* operation is also defined, allowing an operation in progress to be canceled.

Each X.500 operation and result can be *signed* to ensure its integrity. Signing is done using the originating client's or server's public key. The signed request or result is carried end-to-end in the protocol, allowing integrity to be checked at every step. This guards against connection hijacking or modification by intermediate servers. *Service controls* can be specified that determine information such as how an operation will be carried out, whether aliases should be dereferenced, the maximum number of entries to return, and the maximum amount of time to spend on an operation.

In X.500, the directory is distributed among many servers (called DSAs for Directory

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

System Agent). No matter which server a client connects to, it sees the same view of the directory. If a server is unable to answer a client's request, it can either *chain* the request to another server, or *refer* the client to the server.

3. Overview of LDAP

LDAP assumes the same information model and namespace as X.500. It is also client-server based, with one important difference: there are no referrals returned in LDAP. An LDAP server must return only results or errors to a client. If referrals are involved, the LDAP server is responsible for chasing them down. This model is depicted in Figure 2, though the intermediate server shown is not required (i.e., an implementation could choose to have its DSA speak "native" LDAP).

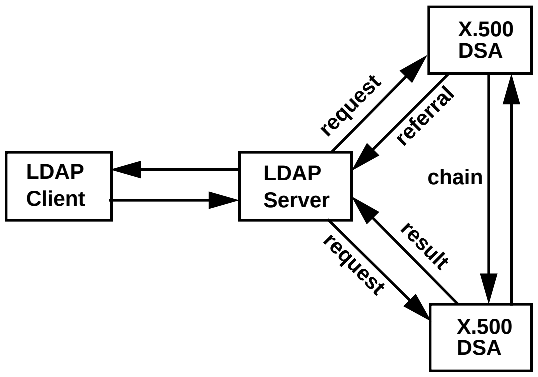

Figure 2. Relationship between LDAP and X.500. The LDAP client-server model includes an LDAP server translating LDAP requests into X.500 requests, chasing X.500 referrals, and returning results to the client.

The LDAP functional model is a subset of the X.500 model. LDAP supports the following operations: search, add, delete, modify, modify RDN, bind, unbind, and abandon. The search operation is similar to its DAP counterpart. A base object and scope are specified, determining which portion of the tree to search. A filter specifies the entries within the scope to select. The LDAP search filter offers the same functionality as the one in DAP but is encoded in a simpler form. The time and size limit service controls are included directly in the search request. (They are not included with the other operations.) The *searchAliases* search control and *dereferenceAliases* service control are combined in a single *derefAliases* parameter in the LDAP search. The ASN.1 [11] definition of the LDAP search request is shown in Figure 3.

```
SearchRequest ::= [APPLICATION 3] SEQUENCE {
 baseObject  LDAPDN,
 scope       ENUMERATED {
  baseObject   (0),
  singleLevel  (1),
  wholeSubtree (2)
 },
 derefAliasesENUMERATED   {
    neverDerefAliases     (0),
    derefInSearching      (1),
    derefFindingBaseObj   (2),
    alwaysDerefAliases    (3)
 },
 sizeLimit     INTEGER (0 .. MaxInt),
 timeLimit     INTEGER (0 .. MaxInt),
 attrsOnly     BOOLEAN,
 filter        Filter,
 attributes    SEQUENCE OF AttributeType
}
Filter ::= CHOICE {
   and            [0] SET OF Filter,
   or             [1] SET OF Filter,
   not            [2] Filter,
   equalityMatch  [3] AttributeValueAssertion,
   substrings     [4] SubstringFilter,
   greaterOrEqual [5] AttributeValueAssertion,
   lessOrEqual    [6] AttributeValueAssertion,
   present        [7] AttributeType,
   approxMatch    [8] AttributeValueAssertion
}
```

Figure 3. ASN.1 for the LDAP search operation. The LDAP search operation offers similar functionality to DAP search. It combines search parameters and service controls and simplifies the filter encoding.

The *LDAPDN* and *AttributeType* components of the search are encoded as simple character strings using the formats defined in RFC 1779 [5] and RFC 1778 [2], respectively, rather than the highly structured encoding used by X.500. Similarly, the value in an *AttributeValueAssertion* is encoded as a primitive OCTETSTRING, not a more structured ASN.1 type. The structure is reflected in the syntax of the encoded string, not in the encoding itself.

The results of an LDAP search are sent back to the client one at a time, in separate *searchEntry* packets. This sequence of entries is terminated by a final *searchResult* packet that contains the result of the search (e.g., suc-

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.