ForgeRock Directory Services

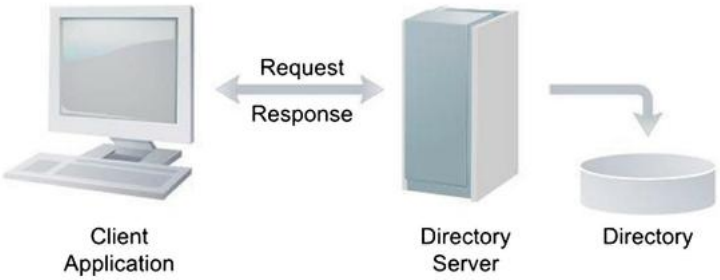
Directory and Directory Server:

We all deal with directories daily. We use a telephone directory to look up phone numbers. When visiting a library, we use the library catalogue to look up the books we want to read.

We use the file system directory with computers to store our files and documents. Simply put, a ***directory is a repository of information***. The information is usually organized so that it can be retrieved easily.

Directories on a network are typically accessed using the client/server communication model. Applications wanting to read or write data to a directory, communicate with specialized servers. **The directory server performs a read or write operation on the actual directory**.

Figure below shows this Directory server and client interaction



The communication between the directory server and client applications is usually accomplished using standardized protocols. **The Lightweight Directory Access Protocol (LDAP)** provides a standard protocol for communicating with a directory. The directory servers that implement the LDAP protocol are usually called LDAP servers.

The LDAP protocol is based on an earlier X.5001 standard but is significantly simpler (lightweight) and easily extensible. Over the years, the LDAP protocol went through iterations and is currently at version 3.0.

LDAP History

LDAP was developed by Tim Howes, Steve Kille, and Wengyik Yeong to create a network protocol to get data out. In 1993 appeared the first draft of the RFC 1487,17 which contained the specification of LDAP based on the access of X.500.(ref: https://fastercapital.com/content/X-500--OIDs-in-the-World-of-Directory-Services.html)

The first version acts as a proxy or gateway to X.500 directories, a comprehensive directory service developed in the 1980s.

Tim Howes, with his colleagues, created the Open Source University of Michigan LDAP Implementation, which became the reference for all the LDAP servers. The project's website is active and accessible only for historical reference.

The second version, the first to be operative (LDAPv2), was released in 1993 as an Internet Engineering Task Force (IETF) Proposed Standard with basic operations like searching and modifying information.

ForgeRock Directory Services

This version offers limited functionality and has some problems related to the security mechanisms. The third version, the latest available, was released in 1997, including many improvements related to security, like Transport Layer Security (TLS) and the possibility of supporting referrals.

It was based on RFCs like RFC 225119 and RFC 4519,20 which explain the protocol and the supported data models. This version became the de facto standard for directory services, and many applications have support to integrate and use LDAP to obtain certain information.

LDAP Vendors

LDAP gained wide support from various vendors. There has also been a strong open source movement to produce LDAP servers. Table 1-5 outlines some of the popular directory servers and include the information about which is the docker image to run each of them with a small configuration.

Directory Name	Vendor	Open Source?	URL	Docker Image
Apache DS	Apache	Yes	https://directory.apache.org/apacheds/	https://hub.docker.com/r/openmicroscopy/apacheds
OpenLDAP	OpenLDAP	Yes	https://www.openldap.org/	https://hub.docker.com/r/bitnami/openldap
Tivoli Directory Server	IBM	No	https://www.ibm.com/docs/en/i/7.5?topic=services-tivoli-directory-server-i-ldap	No existing official image; you need to create one manually.
Active Directory	Microsoft	No	https://learn.microsoft.com/en-us/windows/win32/adsi/so-what-is-active-directory?redirectedfrom=MSDN	No existing official image; you need to create one manually.
eDirectory	Novell	No	https://www.microfocus.com/en-us/cyberres/identity-access-management/edirectory	No existing official image; you must create one manually.
Oracle Directory Server Enterprise Edition (ODSEE)	Oracle (formerly Sun)	No	https://www.oracle.com/security/identity-management/technologies/directory-server-enterprise-edition	No existing official image; you need to create one manually.
Oracle Internet Directory (OID)		No	https://www.oracle.com/middleware/technologies/internet-directory.html	No existing official image; you need to create one manually.
OpenDJ	ForgeRock	Yes	https://github.com/OpenIdentityPlatform/OpenDJ	https://hub.docker.com/r/openidentityplatform/opendj

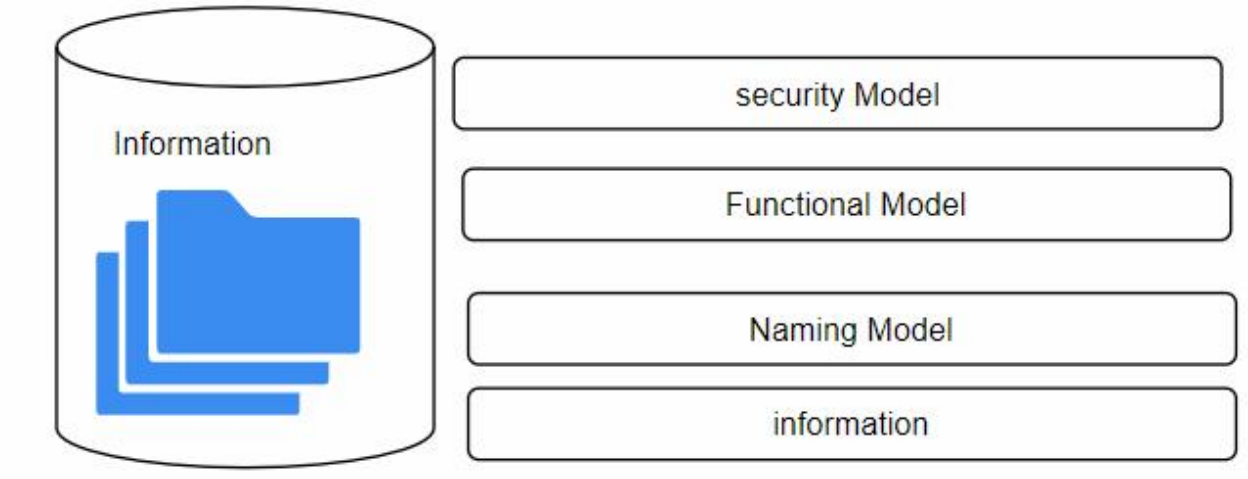
ApacheDS and OpenDJ are pure Java implementations of LDAP directories.

LDAP Overview

LDAP defines a standard protocol used by directory clients and directory servers. LDAP can be better understood by considering the following four models it is based on:

1. The **Information model** determines the structure of information stored in the directory.

2. The **Naming model** defines how information is organized and identified in the directory.
3. The **Functional model** defines the operations performed on the directory.
4. The **Security model** defines how to protect information from unauthorized access.



Directory vs. Database

- Like a database, an LDAP directory stores information. However, several key characteristics set a directory apart from relational databases.

LDAP directories typically store information that is relatively not frequently change. For example, employee information stored in LDAP, such as their phone number or name, does not change daily. However, users and applications look up this information very frequently.

Since the data in a directory is accessed more often than updated, LDAP directories follow the WORM(**W**rite **O**nce and **R**ead **M**any) principle and are heavily optimized for read performance. Relational databases employ referential integrity and locking techniques to ensure data consistency.

The type of data stored in LDAP usually does not warrant such strict consistency requirements. Hence, most of these features need to be present on LDAP servers. Also, transactional semantics to roll back transactions are not defined under LDAP specification.

Relational databases follow normalization principles to avoid data duplication and redundancy. On the other hand, LDAP directories are organized in a hierarchical, object-oriented way. This organization violates some of the normalization principles. Also, there needs to be a concept of table joins in LDAP. Even though directories lack several of the RDBMS features mentioned earlier, many modern LDAP directories are built on top of relational databases such as DB2,5 MySQL and PostgreSQL.

At some point, LDAP has similar characteristics to nonrelational databases like Cassandra, MongoDB, and many others where the performance of the write/read, high availability, and scalability are more relevant than the consistency.

Information Model

The basic unit of information stored in LDAP is an entry. Entries hold information about real-world objects such as employees, servers, printers, and organizations. Each entry in an LDAP directory comprises zero or more attributes. Attributes are key-value pairs that hold information about the object the entry represents.

The key portion of an attribute is also called the attribute type and describes the information that can be stored in the attribute. The value portion of the attribute contains the actual information. Below table shows a portion of an entry representing an employee. The left column in the entry contains the attribute types, and the right column holds the attribute values.

Table 1-1 Employee LDAP Entry

Employee Entry	
objectClass	inetOrgPerson
givenName	John
surname	Smith
mail	john@inflix.com jsmith@inflix.com
mobile	+1 801 100 1000

Note:

Attribute names, by default, are case-insensitive. However, it is recommended to use camel case format in LDAP operations.

We will notice that the mail attribute has two values. **Attributes that are allowed to hold multiple values are called multivalued attributes**. Single-valued attributes, on the other hand, can only hold a single value. **The LDAP specification does not guarantee the order of the values in a multivalued attribute**.

Each attribute type is associated with a syntax that dictates the format of the data stored as an attribute value. For example, the mobile attribute type has a telephoneNumber syntax. This forces the attribute to hold a string value with a length between 1 and 32.

Additionally, the syntax also defines the attribute value behavior during search operations. For example, the givenName attribute has the syntax DirectoryString. This syntax enforces that only alphanumeric characters are allowed as values. Table 1-2 lists some common attributes and their associated syntax description.

Table 1-2 Common Entry Attributes

Attribute Type	Syntax	Description
commonName	DirectoryString	Stores the common name of a person.
company	DirectoryString	Stores the company's name.
employeeNumber	DirectoryString	Stores the employee's identification number in the organization.
givenName	DirectoryString	Stores the user's first name.
jpegPhoto	Binary	Stores one or more images of the person.
mail	IA5 String	Stores a person's SMTP mail address.
mobile	TelephoneNumber	Stores a person's mobile number.
postalAddress	Postal Address	Stores the user's ZIP or postal code.
postalCode	DirectoryString	Stores the user's ZIP or postal code.
st	DirectoryString	Stores the state or province name.
street	DirectoryString	Stores the street address.
surname	DirectoryString	Stores the last name of the person.
telephoneNumber	TelephoneNumber	Stores the person's primary telephone number.
uid	DirectoryString	Stores the user id.
title	DirectoryString	Stores the name of the position or the company's function.
wwwhomepage	DirectoryString	Stores the official web page of the company.

The attributes in Table 1-2 are the most used for the developers and tools. Still, there is a big list of other attributes depending on whether vendor or the tool supports it or not; for example, on the official web page of Microsoft, all the attributes are supported for the Active Directory.

Object Classes

In object-oriented languages, such as Java, we create a class and use it as a blueprint for creating objects. The class defines the attributes/data (and behaviour/methods) that these instances can have.

Similarly, ***object classes in LDAP determine the attributes an LDAP entry can have***. These object classes also define which attributes are mandatory and which are optional.

Every LDAP entry has a special attribute aptly named objectClass that holds the object class it belongs to. Looking at the objectClass value in the employee entry in Table 1-1, we can conclude that the entry belongs to the inetOrgPerson class.

Table 1-3 below shows the required and optional attributes in a standard LDAP person object class. The cn attribute holds the person's common name, whereas the sn attribute holds the person's family name or surname.

Table 1-3 Person Object Class

Required Attributes	Optional Attributes
sn	description
	telephoneNumber
cn	userPassword
objectClass	seeAlso

As in Java, an object class can extend other object classes. This inheritance will allow the child object class to inherit parent class attributes. For example, the person object class defines attributes such as common name and surname.

The object class inetOrgPerson extends the person class and thus inherits all the person's attributes. Additionally, inetOrgPerson defines attributes required for a person working in an organization, such as departmentNumber and employeeNumber.

One special object class, namely, top, does not have any parents. All other object classes are childs of top and inherit all the attributes declared in it. The top object class includes the mandatory objectClass attribute. Figure 1-2 shows the object inheritance.

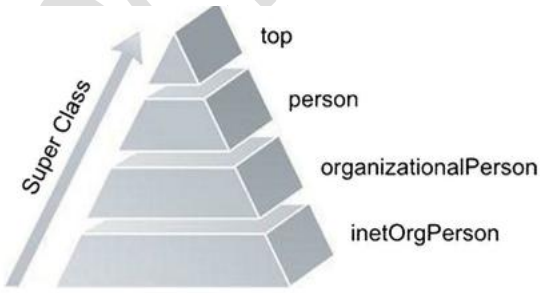


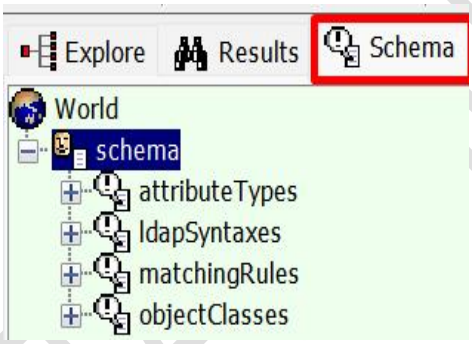
Figure 1-2 LDAP object inheritance

Most LDAP implementations come with standard object classes that can be used out of the box. The below Table 1-4 lists some of these LDAP object classes and their commonly used attributes.

Object Class	Attributes	Description
top	objectClass	Defines the root object class. All other object classes must extend this class.
organization	o	Represents a company or an organization. The o attribute typically holds the name of the organization.
organizationalUnit	ou	Represents a department or similar entity inside an organization.
person	sn cn telephoneNumber userPassword	Represents a person in the directory and requires the sn (surname) and cn (common name) attributes.
organizationalPerson	registeredAddress postalAddress postalCode	Subclasses that represent a person in an organization.
inetOrgPerson	uid departmentNumber employeeNumber givenName manager	It provides additional attributes and can be used to represent a person working in today's Internet- and intranet-based organization. The uid attribute holds the person's username or user id.

Directory Schema

The LDAP directory schema is a set of rules determining the type of information stored in a directory. Schemas can be considered packaging units and contain attribute type definitions and object class definitions.



The schema rules are verified before an entry can be stored in LDAP. This schema checking ensures that the entry has all the required attributes and contains no attributes not part of the schema. The below Figure 1-3 represents a generic LDAP schema.

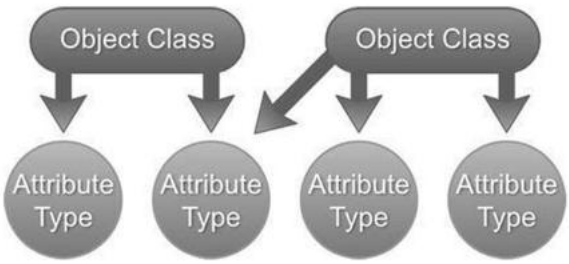


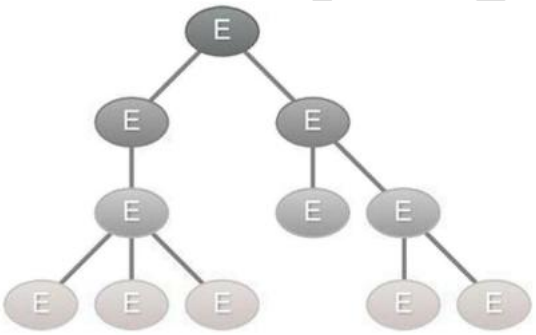
Figure 1-3 LDAP generic schema

Like databases, directory schemas must be well designed to address issues like data redundancy. Before implementing our schema, it is worth looking at publicly available standard schemas. These standard schemas often contain all definitions to store the required data and, more importantly, ensure interoperability across other directories.

Naming Model:

The LDAP Naming model defines how entries are organized in a directory. It also determines how a particular entry can be uniquely identified. The Naming model recommends that entries be stored logically in a hierarchical fashion. This entry tree is often called a ***directory information tree (DIT).***

Below Figure 1-4 provides an example of a generic directory tree.



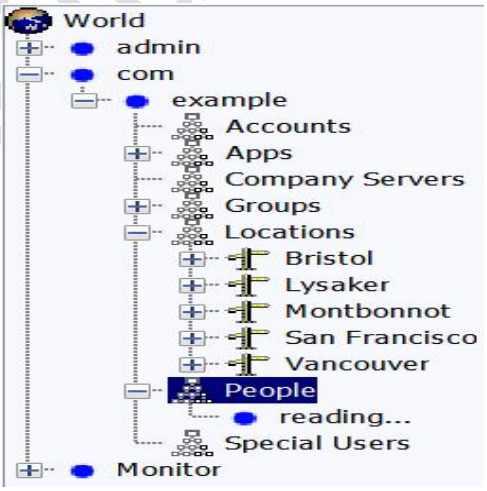


Figure 1-4 Generic DIT

The tree's root is usually referred to as the base or suffix of the directory. This entry represents the organization that owns the directory. The format of suffixes can vary from implementation to implementation, but generally, there are three recommended approaches, as listed in below Figure 1-5.

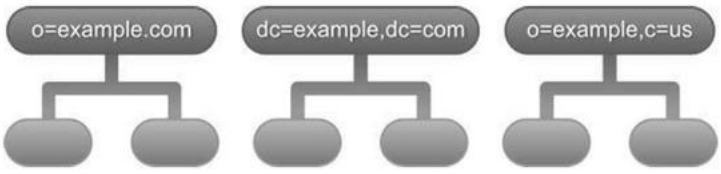


Figure 1-5 Directory suffix naming conventions

Note: DC stands for domain component.

- ✓ ***The first recommended technique is to use the organization's domain name as the suffix***. For example, if the organization's domain name is example.com, the directory suffix will be o=example.com.

- ✓ The second technique also uses the domain name, but each name component is prepended with "dc=" and joined by commas. So, the domain name example.com would result in a suffix dc=example, dc=com. This technique is proposed in RFC 224714 and is popular with **Microsoft Active Directory.**

- ✓ The third technique uses the X.500 model and creates a suffix in the format o=organization name, c=country code. In the United States, the suffix for the organization example would be o=example, c=us.

The naming model also defines naming and uniquely identifying entries in a directory. Entries with a common immediate parent are uniquely identified via their ***relative distinguished name (RDN), also called distinguished name (DN)***.

The RDN is computed using one or more attribute/value pairs of the entry. In its simplest case, RDN is usually of the form **attribute-name = attribute value**.

Figure 1-6 below provides a simplified representation of an organization directory. Each person entry under ou=employees has a unique uid. So the RDN for the first person entry would be uid=emp1, where emp1 is the employee's user id.

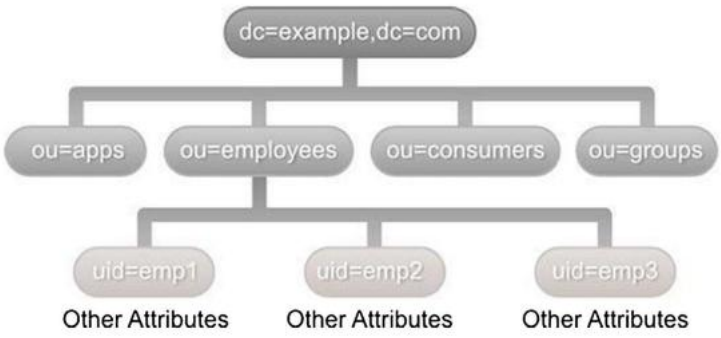


Figure 1-6 Example of an organization directory

Note: The distinguished name is not an actual attribute in the entry. It is a logical name associated with the entry.

It is important to remember that RDN cannot be used to uniquely identify the entry in the entire tree. However, this can be easily done by combining the RDNs of all the entries in the path from the top of the tree to the entry. The result of this combination is referred to as distinguished name (DN). In above Figure 1-6, the DN for person 1 would be uid=emp1, ou=employees, dc=example, dc=com.

Since the DN is made by combining RDNs, if an entry's RDN changes, the DNs of that entry and all its child entries also change. There can be situations where a set of entries does not have a unique attribute. In those scenarios, one option is to combine multiple attributes to create uniqueness.

For example, we can use the consumer's common name and email address in the previous directory as an RDN. Multivalued RDNs are represented by separating each attribute pair with a +, like so:

cn = Narsi + mail=narsi@inflinx.com

Some special characters on the RDN must be escaped to prevent different problems. The special characters are + (plus), = (equals), < (less than), > (greater than), ; (semicolon), , (comma), \ (backslash), # (number sign), and ". There are different approaches to escape the characters, like adding the backslash ("\" ASCII 92) before the special character; this approach is the most commonly used. The other replaces the special characters with a backslash and hexadecimal digit, and the last one surrounds the attribute's value with quotation marks ("").

Note Multivalued RDNs are usually discouraged. Creating a unique sequence attribute is recommended in those scenarios to ensure uniqueness.

Functional Model

The LDAP Functional model describes the access and modification operations that can be performed on the directory using the LDAP protocol. These operations fall into three categories: ***query, update, and authentication***.

The query operations are used to search and retrieve information from a directory. So whenever some information needs to be read, a search query must be constructed and executed against LDAP.

The search operation takes a starting point within DIT, the search depth, and the attributes an entry must have for a match.

The update operations add, modify, delete, and rename directory entries. As the name suggests, the add operation adds a new entry to the directory. This operation requires the DN of the entry to be created and a set of attributes that constitute the entry.

The delete operation takes a fully qualified DN of the entry and deletes it from the directory. The LDAP protocol allows only the leaf entries to be deleted.

The modified operation updates an existing entry. This operation takes the entry's DN and a set of modifications, such as adding a new attribute, updating a new one, or removing an existing one.

The rename operation can rename or move entries in a directory.

The authentication operations are used for connecting and ending sessions between the client and the LDAP server. A bind operation initiates an LDAP session between the client and server. Typically, this would result in an anonymous session.

The client can provide a DN and credentials to authenticate and create an authenticated session. On the other hand, the unbind operation can be used to terminate an existing session and disconnect from the server.

LDAP V3 introduced a framework for extending existing operations and adding new ones without changing the protocol.

ForgeRock Directory Services

Security Model

The LDAP Security model protects LDAP directory information from unauthorized access. The model specifies which clients can access which parts of the directory and what kinds of operations (search vs. update) are allowed.

The LDAP Security model is based on the client authenticating to the server. As discussed earlier, this authentication process or bind operation involves the client supplying a DN identifying itself and a password.

An anonymous session is established if the client does not provide a DN and password. RFC 282915 defines a set of authentication methods that LDAP V3 servers must support.

After successful authentication, the access control models are consulted to determine whether the client has sufficient privileges to do what is requested. Unfortunately, no standards exist for access control models, and each vendor provides their implementations.

Sample Application

Lets work with a directory for a hypothetical book library. Lets chosen the library because the concept is universal and easy to grasp. A library usually stores books and other multimedia that patrons can borrow.

Libraries also employ people to take care of daily library operations. To keep things manageable, the directory will not store book information. A relational database is suitable for recording book information. The below Figure 1-7 shows the LDAP directory tree for our library application.