OpenLDAP Documentation

Release 2.4.42

OpenLDAP Foundation

May 10, 2017
# Contents

1 Preface                                                                                           1
  1.1 Preface ................................................................. 2
  1.2 Introduction to OpenLDAP Directory Services ........................................................................... 3
  1.3 Quick Start ................................................................. 11
  1.4 The Big Picture ........................................................... 15
  1.5 Building and Installing OpenLDAP Software .............................................................................. 16
  1.6 Configuring slapd ......................................................... 16
  1.7 Running slapd ............................................................... 16
  1.8 Access Control ............................................................. 16
  1.9 Database Creation and Maintenance Tools .................................................................................. 16
  1.10 Backends ........................................................................ 16
  1.11 Schema Specification .................................................... 16
  1.12 Security Considerations .................................................. 16
  1.13 Using SASL ................................................................. 16
  1.14 Using TLS ....................................................................... 16
  1.15 OpenLDAP Software Copyright Notices .................................................................................... 16
  1.16 OpenLDAP Public License ............................................. 18
  1.17 Glossary ......................................................................... 18

2 Indices and tables  ..................................................................... 19
Preface

This document is considered a part of OpenLDAP Software. This document is subject to terms of conditions set forth in OpenLDAP Software Copyright Notices and the OpenLDAP Public License. Complete copies of the notices and associated license can be found in Appendix K and L, respectively.

Portions of OpenLDAP Software and this document may be copyright by other parties and/or subject to additional restrictions. Individual source files should be consulted for additional copyright notices.

Scope of this Document

This document provides a guide for installing OpenLDAP Software on Unix (and UNIX-like) systems. The document is aimed at experienced system administrators with basic understanding of LDAP-based directory services.

This document is meant to be used in conjunction with other ldapdif information resources provided with the software package and on the project’s internet site (http://www.openldap.org/).

The site makes available a number of resources:

Document Catalog  http://www.openldap.org/doc/
Frequently Asked Questions  http://www.openldap.org/faq/
Issue Tracking System  http://www.openldap.org/its/
Mailing Lists  http://www.openldap.org/lists/
Software Pages  http://www.openldap.org/software/
Support Pages  http://www.openldap.org/support/

This document is not a complete reference for OpenLDAP software; the manual pages are the definitive documentation. For best results, you should use the manual pages that were installed on your system with your version of OpenLDAP software so that you’re looking at documentation that matches the code. While the OpenLDAP web site also provides the manual pages for convenience, you can not assume that they correspond to the particular version you’re running.
Acknowledgements

The OpenLDAP Project is comprised of a team of volunteers. This document would not be possible without their contribution of time and energy.

The OpenLDAP Project would also like to thank the University of Michigan LDAP Team for building the foundation of LDAP software and information to which OpenLDAP Software is built upon. This document is based upon University of Michigan document: The SLAPD and SLURPD Administrators Guide.

Amendments

Suggested enhancements and corrections to this document should be submitted using the OpenLDAP Issue Tracking System (http://www.openldap.org/its/).

Contents:

Preface

This document is considered a part of OpenLDAP Software. This document is subject to terms of conditions set forth in OpenLDAP Software Copyright Notices and the OpenLDAP Public License. Complete copies of the notices and associated license can be found in Appendix K and L, respectively.

Portions of OpenLDAP Software and this document may be copyright by other parties and/or subject to additional restrictions. Individual source files should be consulted for additional copyright notices.

Scope of this Document

This document provides a guide for installing OpenLDAP Software on Unix (and UNIX-like) systems. The document is aimed at experienced system administrators with basic understanding of LDAP-based directory services.

This document is meant to be used in conjunction with other openldap information resources provided with the software package and on the project’s internet site (http://www.openldap.org/).

The site makes available a number of resources:

Document Catalog  http://www.openldap.org/doc/
Frequently Asked Questions  http://www.openldap.org/faq/
Issue Tracking System  http://www.openldap.org/its/
Mailing Lists  http://www.openldap.org/lists/
Software Pages  http://www.openldap.org/software/
Support Pages  http://www.openldap.org/support/

This document is not a complete reference for OpenLDAP software; the manual pages are the definitive documentation. For best results, you should use the manual pages that were installed on your system with your version of OpenLDAP software so that you’re looking at documentation that matches the code. While the OpenLDAP web site also provides the manual pages for convenience, you can not assume that they correspond to the particular version you’re running.
Acknowledgements

The OpenLDAP Project is comprised of a team of volunteers. This document would not be possible without their contribution of time and energy.

The OpenLDAP Project would also like to thank the University of Michigan LDAP Team for building the foundation of LDAP software and information to which OpenLDAP Software is built upon. This document is based upon University of Michigan document: The SLAPD and SLURPD Administrators Guide.

Amendments

Suggested enhancements and corrections to this document should be submitted using the OpenLDAP Issue Tracking System (http://www.openldap.org/its/).

Introduction to OpenLDAP Directory Services

This document describes how to build, configure, and operate OpenLDAP Software to provide directory services. This includes details on how to configure and run the Standalone LDAP Daemon, slapd (8). It is intended for new and experienced administrators alike. This section provides a basic introduction to directory services and, in particular, the directory services provided by slapd (8). This introduction is only intended to provide enough information so one might get started learning about LDAP, X.500, and directory services.

What is a directory service?

A directory is a specialized database specifically designed for searching and browsing, in addition to supporting basic lookup and update functions.

| Note: | A directory is defined by some as merely a database optimized for read access. This definition, at best, is overly simplistic. |

Directories tend to contain descriptive, attribute-based information and support sophisticated filtering capabilities. Directories generally do not support complicated transaction or roll-back schemes found in database management systems designed for handling high-volume complex updates. Directory updates are typically simple all-or-nothing changes, if they are allowed at all. Directories are generally tuned to give quick response to high-volume lookup or search operations. They may have the ability to replicate information widely in order to increase availability and reliability, while reducing response time. When directory information is replicated, temporary inconsistencies between the replicas may be okay, as long as inconsistencies are resolved in a timely manner.

There are many different ways to provide a directory service. Different methods allow different kinds of information to be stored in the directory, place different requirements on how that information can be referenced, queried and updated, how it is protected from unauthorized access, etc. Some directory services are local, providing service to a restricted context (e.g., the finger service on a single machine). Other services are global, providing service to a much broader context (e.g., the entire Internet). Global services are usually distributed, meaning that the data they contain is spread across many machines, all of which cooperate to provide the directory service. Typically a global service defines a uniform namespace which gives the same view of the data no matter where you are in relation to the data itself.

A web directory, such as provided by the Open Directory Project, is a good example of a directory service. These services catalog web pages and are specifically designed to support browsing and searching.
While some consider the Internet Domain Name Service (DNS) is an example of a globally distributed directory service, DNS is not browseable nor searchable. It is more properly described as a globally distributed lookup service.

What is LDAP?

LDAP stands for Lightweight Directory Access Protocol. As the name suggests, it is a lightweight protocol for accessing directory services, specifically X.500-based directory services. LDAP runs over TCP/IP or other connection oriented transfer services. LDAP is an IETF Standard Track protocol and is specified in “Lightweight Directory Access Protocol (LDAP) Technical Specification Road Map” RFC4510.

This section gives an overview of LDAP from a user’s perspective.

What kind of information can be stored in the directory?

The LDAP information model is based on entries. An entry is a collection of attributes that has a globally-unique Distinguished Name (DN). The DN is used to refer to the entry unambiguously. Each of the entry’s attributes has a type and one or more values. The types are typically mnemonic strings, like cn for common name, or mail for email address. The syntax of values depend on the attribute type. For example, a cn attribute might contain the value “Babs Jensen”. mail attribute might contain the value babs@example.com. A jpegPhoto attribute would contain a photograph in the JPEG (binary) format.

How is the information arranged?

In LDAP, directory entries are arranged in a hierarchical tree-like structure. Traditionally, this structure reflected the geographic and/or organizational boundaries. Entries representing countries appear at the top of the tree. Below them are entries representing states and national organizations. Below them might be entries representing organizational units, people, printers, documents, or just about anything else you can think of. Fig. 1.1 shows an example LDAP directory tree using traditional naming.

The tree may also be arranged based upon Internet domain names. This naming approach is becoming increasingly popular as it allows for directory services to be located using the DNS. Fig. 1.2 shows an example LDAP directory tree using domain-based naming.

In addition, LDAP allows you to control which attributes are required and allowed in an entry through the use of a special attribute called objectClass. The values of the objectClass attribute determine the schema rules the entry must obey.

How is the information referenced?

An entry is referenced by its distinguished name, which is constructed by taking the name of the entry itself (called the Relative Distinguished Name or RDN) and concatenating the names of its ancestor entries. For example, the entry for Barbara Jensen in the Internet naming example above has an RDN of uid=babs and a DN of uid=babs, ou=People, dc=example, dc=com. The full DN format is described in RFC4514, “LDAP: String Representation of Distinguished Names.”

How is the information accessed?

LDAP defines operations for interrogating and updating the directory. Operations are provided for adding and deleting an entry from the directory, changing an existing entry, and changing the name of an entry. Most of the time, though, LDAP is used to search for information in the directory. The LDAP search operation allows some portion of the
Fig. 1.1: LDAP directory tree (traditional naming)
Fig. 1.2: LDAP directory tree (Internet naming)
directory to be searched for entries that match some criteria specified by a search filter. Information can be requested from each entry that matches the criteria.

For example, you might want to search the entire directory subtree at and below \texttt{dc=example,dc=com} for people with the name “Barbara Jensen”, retrieving the email address of each entry found. LDAP lets you do this easily. Or you might want to search the entries directly below the \texttt{st=California,c=US} entry for organizations with the string \textit{Acme} in their name, and that have a fax number. LDAP lets you do this too. The next section describes in more detail what you can do with LDAP and how it might be useful to you.

**How is the information protected from unauthorized access?**

Some directory services provide no protection, allowing anyone to see the information. LDAP provides a mechanism for a client to authenticate, or prove its identity to a directory server, paving the way for rich access control to protect the information the server contains. LDAP also supports data security (integrity and confidentiality) services.

**When should I use LDAP?**

This is a very good question. In general, you should use a Directory server when you require data to be centrally managed, stored and accessible via standards based methods.

Some common examples found throughout the industry are, but not limited to:

- Machine Authentication
- User Authentication
- User/System Groups
- Address book
- Organization Representation
- Asset Tracking
- Telephony Information Store
- User resource management
- E-mail address lookups
- Application Configuration store
- PBX Configuration store
- etc…

There are various Distributed Schema Files that are standards based, but you can always create your own Schema Specification.

There are always new ways to use a Directory and apply LDAP principles to address certain problems, therefore there is no simple answer to this question.

If in doubt, join the general LDAP forum for non-commercial discussions and information relating to LDAP at: http://www.umich.edu/~dirsvcs/ldap/mailinglist.html and ask!

**When should I not use LDAP?**

When you start finding yourself bending the directory to do what you require, maybe a redesign is needed. Or if you only require one application to use and manipulate your data (for discussion of LDAP vs RDBMS, please read the LDAP vs RDBMS section).
It will become obvious when LDAP is the right tool for the job.

**How does LDAP work?**

LDAP utilizes a client-server model. One or more LDAP servers contain the data making up the directory information tree (*DIT*). The client connects to servers and asks it a question. The server responds with an answer and/or with a pointer to where the client can get additional information (typically, another LDAP server). No matter which LDAP server a client connects to, it sees the same view of the directory; a name presented to one LDAP server references the same entry it would at another LDAP server. This is an important feature of a global directory service.

**What about X.500?**

Technically, LDAP is a directory access protocol to an X.500 directory service, the OSI directory service. Initially, LDAP clients accessed gateways to the X.500 directory service. This gateway ran LDAP between the client and gateway and X.500’s Directory Access Protocol (DAP) between the gateway and the X.500 server. DAP is a heavyweight protocol that operates over a full OSI protocol stack and requires a significant amount of computing resources. LDAP is designed to operate over TCP/IP and provides most of the functionality of DAP at a much lower cost.

While LDAP is still used to access X.500 directory service via gateways, LDAP is now more commonly directly implemented in X.500 servers.

The Standalone LDAP Daemon, or *slapd* (8), can be viewed as a lightweight X.500 directory server. That is, it does not implement the X.500’s DAP nor does it support the complete X.500 models.

If you are already running a X.500 DAP service and you want to continue to do so, you can probably stop reading this guide. This guide is all about running LDAP via *slapd* (8), without running X.500 DAP. If you are not running X.500 DAP, want to stop running X.500 DAP, or have no immediate plans to run X.500 DAP, read on.

It is possible to replicate data from an LDAP directory server to a X.500 DAP **DSA**. This requires an LDAP/DAP gateway. OpenLDAP Software does not include such a gateway.

**What is the difference between LDAPv2 and LDAPv3?**

LDAPv3 was developed in the late 1990’s to replace LDAPv2.

LDAPv3 adds the following features to LDAP:

- Strong authentication and data security services via **SASL**
- Certificate authentication and data security services via **TLS**
- Internationalization through the use of Unicode
- Referrals and Continuations
- Schema Discovery
- Extensibility (controls, extended operations, and more)

LDAPv2 is historic (RFC3494). As most so-called LDAPv2 implementations (including *slapd* (8)) do not conform to the LDAPv2 technical specification, interoperability amongst implementations claiming LDAPv2 support is limited. As LDAPv2 differs significantly from LDAPv3, deploying both LDAPv2 and LDAPv3 simultaneously is quite problematic. LDAPv2 should be avoided. LDAPv2 is disabled by default.
LDAP vs RDBMS

This question is raised many times, in different forms. The most common, however, is: Why doesn’t OpenLDAP drop Berkeley DB and use a relational database management system (RDBMS) instead? In general, expecting that the sophisticated algorithms implemented by commercial-grade RDBMS would make OpenLDAP be faster or somehow better and, at the same time, permitting sharing of data with other applications.

The short answer is that use of an embedded database and custom indexing system allows OpenLDAP to provide greater performance and scalability without loss of reliability. OpenLDAP uses Berkeley DB concurrent / transactional database software. This is the same software used by leading commercial directory software.

Now for the long answer. We are all confronted all the time with the choice RDBMSes vs. directories. It is a hard choice and no simple answer exists.

It is tempting to think that having a RDBMS backend to the directory solves all problems. However, it is a pig. This is because the data models are very different. Representing directory data with a relational database is going to require splitting data into multiple tables.

Think for a moment about the person objectclass. Its definition requires attribute types objectclass, sn and cn and allows attribute types userPassword, telephoneNumber, seeAlso and description. All of these attributes are multivalued, so a normalization requires putting each attribute type in a separate table.

Now you have to decide on appropriate keys for those tables. The primary key might be a combination of the DN, but this becomes rather inefficient on most database implementations.

The big problem now is that accessing data from one entry requires seeking on different disk areas. On some applications this may be OK but in many applications performance suffers.

The only attribute types that can be put in the main table entry are those that are mandatory and single-value. You may add also the optional single-valued attributes and set them to NULL or something if not present.

But wait, the entry can have multiple objectclasses and they are organized in an inheritance hierarchy. An entry of objectclass organizationalPerson now has the attributes from person plus a few others and some formerly optional attribute types are now mandatory.

What to do? Should we have different tables for the different objectclasses? This way the person would have an entry on the person table, another on organizationalPerson, etc. Or should we get rid of person and put everything on the second table?

But what do we do with a filter like \( (cn=*) \) where cn is an attribute type that appears in many, many objectclasses. Should we search all possible tables for matching entries? Not very attractive.

Once this point is reached, three approaches come to mind. One is to do full normalization so that each attribute type, no matter what, has its own separate table. The simplistic approach where the DN is part of the primary key is extremely wasteful, and calls for an approach where the entry has a unique numeric id that is used instead for the keys and a main table that maps DNs to ids. The approach, anyway, is very inefficient when several attribute types from one or more entries are requested. Such a database, though cumbersomely, can be managed from SQL applications.

The second approach is to put the whole entry as a blob in a table shared by all entries regardless of the objectclass and have additional tables that act as indices for the first table. Index tables are not database indices, but are fully managed by the LDAP server-side implementation. However, the database becomes unusable from SQL. And, thus, a fully fledged database system provides little or no advantage. The full generality of the database is unneeded. Much better to use something light and fast, like Berkeley DB.

A completely different way to see this is to give up any hopes of implementing the directory data model. In this case, LDAP is used as an access protocol to data that provides only superficially the directory data model. For instance, it may be read only or, where updates are allowed, restrictions are applied, such as making single-value attribute types that would allow for multiple values. Or the impossibility to add new objectclasses to an existing entry or remove one of those present. The restrictions span the range from allowed restrictions (that might be elsewhere the result of access
control) to outright violations of the data model. It can be, however, a method to provide LDAP access to preexisting data that is used by other applications. But in the understanding that we don’t really have a “directory”.

Existing commercial LDAP server implementations that use a relational database are either from the first kind or the third. I don’t know of any implementation that uses a relational database to do inefficiently what BDB does efficiently. For those who are interested in “third way” (exposing EXISTING data from RDBMS as LDAP tree, having some limitations compared to classic LDAP model, but making it possible to interoperate between LDAP and SQL applications):

OpenLDAP includes back-sql - the backend that makes it possible. It uses ODBC + additional metainformation about translating LDAP queries to SQL queries in your RDBMS schema, providing different levels of access - from read-only to full access depending on RDBMS you use, and your schema.

For more information on concept and limitations, see slapd-sql (5) man page, or the Backends section. There are also several examples for several RDBMSes in servers/slapd/back-sql/rdbms_depend/* subdirectories.

What is slapd and what can it do?

*slapd* (8) is an LDAP directory server that runs on many different platforms. You can use it to provide a directory service of your very own. Your directory can contain pretty much anything you want to put in it. You can connect it to the global LDAP directory service, or run a service all by yourself. Some of slapd’s more interesting features and capabilities include:

**LDAPv3** slapd implements version 3 of LDAP. slapd supports LDAP over both IPv4 and IPv6 and Unix IPC.

**Simple Authentication and Security Layer** slapd supports strong authentication and data security (integrity and confidentiality) services through the use of SASL. slapd’s SASL implementation utilizes Cyrus SASL software which supports a number of mechanisms including DIGEST-MD5, EXTERNAL, and GSSAPI.

**Transport Layer Security** slapd supports certificate-based authentication and data security (integrity and confidentiality) services through the use of TLS (or SSL). slapd’s TLS implementation can utilize OpenSSL, GnuTLS, or MozNSS software.

**Topology control** slapd can be configured to restrict access at the socket layer based upon network topology information. This feature utilizes TCP wrappers.

**Access control** slapd provides a rich and powerful access control facility, allowing you to control access to the information in your database(s). You can control access to entries based on LDAP authorization information, IP address, domain name and other criteria. slapd supports both static and dynamic access control information.

**Internationalization** slapd supports Unicode and language tags.

**Choice of database backends** slapd comes with a variety of different database backends you can choose from. They include BDB, a high-performance transactional database backend; HDB, a hierarchical high-performance transactional backend; SHELL, a backend interface to arbitrary shell scripts; and PASSWD, a simple backend interface to the passwd (5) file. The BDB and HDB backends utilize Oracle Berkeley DB.

**Multiple database instances** slapd can be configured to serve multiple databases at the same time. This means that a single slapd server can respond to requests for many logically different portions of the LDAP tree, using the same or different database backends.

**Generic modules API** If you require even more customization, slapd lets you write your own modules easily. slapd consists of two distinct parts: a front end that handles protocol communication with LDAP clients; and modules which handle specific tasks such as database operations. Because these two pieces communicate via a well-defined C API, you can write your own customized modules which extend slapd in numerous ways. Also, a number of programmable database modules are provided. These allow you to expose external data sources to slapd using popular programming languages (Perl, shell, and SQL).

**Threads** slapd is threaded for high performance. A single multi-threaded slapd process handles all incoming requests using a pool of threads. This reduces the amount of system overhead required while providing high performance.
Replication  *slapd* can be configured to maintain shadow copies of directory information. This *single-master/multiple-slave* replication scheme is vital in high-volume environments where a single *slapd* installation just doesn’t provide the necessary availability or reliability. For extremely demanding environments where a single point of failure is not acceptable, *multi-master* replication is also available. *slapd* includes support for *LDAP Sync*-based replication.

Proxy Cache  *slapd* can be configured as a caching LDAP proxy service.

Configuration  *slapd* is highly configurable through a single configuration file which allows you to change just about everything you’d ever want to change. Configuration options have reasonable defaults, making your job much easier. Configuration can also be performed dynamically using LDAP itself, which greatly improves manageability.

Quick Start

The following is a quick start guide to OpenLDAP, including the Standalone *LDAP* Daemon, *slapd* (8).

It is meant to walk you through the basic steps needed to install and configure OpenLDAP Software. It should be used in conjunction with the other chapters of this document, manual pages, and other materials provided with the distribution (e.g. the *INSTALL* document) or on the OpenLDAP web site, in particular the OpenLDAP Software *FAQ* (http://www.openldap.org/faq/?file=2).

If you intend to run OpenLDAP Software seriously, you should review all of this document before attempting to install the software.

Note:  This quick start guide does not use strong authentication nor any integrity or confidential protection services. These services are described in other chapters of the OpenLDAP Administrator’s Guide.

Install from Source

Get the software

You can obtain a copy of the software by following the instructions on the OpenLDAP Software download page (http://www.openldap.org/software/download/). It is recommended that new users start with the latest *release*.

Unpack the distribution

Pick a directory for the source to live under, change directory to there, and unpack the distribution using the following commands:

```
  gunzip -c openldap-VERSION.tgz | tar xvfB -
```

then relocate yourself into the distribution directory:

```
  cd openldap-VERSION
```

You’ll have to replace *VERSION* with the version name of the release.
Review documentation

You should now review the COPYRIGHT, LICENSE, README and INSTALL documents provided with the distribution. The COPYRIGHT and LICENSE provide information on acceptable use, copying, and limitation of warranty of OpenLDAP Software.

You should also review other chapters of this document. In particular the Building and Installing OpenLDAP Software chapter provides detailed information on prerequisite software and installation procedures.

Run configure

You will need to run the provided configure script to configure the distribution for building on your system. configure script accepts many command line options that enable or disable optional software features. Usually the defaults are okay, but you may want to change them. To get a complete list of options that configure accepts, use the --help option:

```
./configure --help
```

However, given that you are using this guide, we’ll assume you are brave enough to just let configure determine what’s best:

```
./configure
```

Assuming configure doesn’t dislike your system, you can proceed with building the software. If configure did complain, well, you’ll likely need to go to the Software FAQ Installation section) and/or read the Building and Installing OpenLDAP Software chapter.

Build the software

The next step is to build the software. This step has two parts, first we construct dependencies and then we compile the software:

```
make depend
make
```

Both invocations of make should complete without error.

Test the build

To ensure a correct build, you should run the test suite (it only takes a few minutes):

```
make test
```

Tests which apply to your configuration will run and they should pass. Some tests, such as the replication test, may be skipped.

Install the software

You are now ready to install the software; this usually requires super-user privileges:
su root -c 'make install'

Everything should now be installed under /usr/local (or whatever installation prefix was used by configure).

Edit the configuration file

Use your favorite editor to edit the provided slapd.ldif example (usually installed as /usr/local/etc/openldap/slapd.ldif) to contain a MDB database definition of the form:

```
dn: olcDatabase=mdb,cn=config
objectClass: olcDatabaseConfig
objectClass: olcMdbConfig
olcDatabase: mdb
OlcDbMaxSize: 1073741824
olcSuffix: dc=<MY-DOMAIN>,dc=<COM>
olcRootDN: cn=Manager,dc=<MY-DOMAIN>,dc=<COM>
olcRootPW: secret
olcDbDirectory: /usr/local/var/openldap-data
olcDbIndex: objectClass eq
```

Be sure to replace <MY-DOMAIN> and <COM> with the appropriate domain components of your domain name. For example, for example.com, use:

```
dn: olcDatabase=mdb,cn=config
objectClass: olcDatabaseConfig
objectClass: olcMdbConfig
olcDatabase: mdb
OlcDbMaxSize: 1073741824
olcSuffix: dc=example,dc=com
olcRootDN: cn=Manager,dc=example,dc=com
olcRootPW: secret
olcDbDirectory: /usr/local/var/openldap-data
olcDbIndex: objectClass eq
```

If your domain contains additional components, such as eng.uni.edu.eu, use:

```
dn: olcDatabase=mdb,cn=config
objectClass: olcDatabaseConfig
objectClass: olcMdbConfig
olcDatabase: mdb
OlcDbMaxSize: 1073741824
olcSuffix: dc=eng,dc=uni,dc=edu,dc=eu
olcRootDN: cn=Manager,dc=eng,dc=uni,dc=edu,dc=eu
olcRootPW: secret
olcDbDirectory: /usr/local/var/openldap-data
olcDbIndex: objectClass eq
```

Details regarding configuring slapd (8) can be found in the slapd-config (5) manual page and the Configuring slapd chapter. Note that the specified olcDbDirectory must exist prior to starting slapd (8).

Import the configuration database

You are now ready to import your configuration database for use by slapd (8), by running the command:
su root -c /usr/local/sbin/slapadd -F /usr/local/etc/cn=config -l /usr/local/etc/openldap/slapd.ldif

Start SLAPD

You are now ready to start the Standalone LDAP Daemon, *slapd* (8), by running the command:

```
su root -c /usr/local/libexec/slapd -F /usr/local/etc/cn=config
```

To check to see if the server is running and configured correctly, you can run a search against it with *ldapsearch* (1). By default, *ldapsearch* is installed as `/usr/local/bin/ldapsearch`:

```
ldapsearch -x -b '' -s base '(objectclass=*)' namingContexts
```

Note the use of single quotes around command parameters to prevent special characters from being interpreted by the shell. This should return:

```
dn: namingContexts: dc=example,dc=com
```

Details regarding running *slapd* (8) can be found in the *slapd* (8) manual page and the *Running slapd* chapter.

Add initial entries to your directory

You can use *ldapadd* (1) to add entries to your LDAP directory. *ldapadd* expects input in **LDIF** form. We’ll do it in two steps:

1. create an LDIF file
2. run *ldapadd*

Use your favorite editor and create an LDIF file that contains:

```
dn: dc=<MY-DOMAIN>,dc=<COM>
objectclass: dcObject
objectclass: organization
o: <MY ORGANIZATION>
dc: <MY-DOMAIN>

dn: cn=Manager,dc=<MY-DOMAIN>,dc=<COM>
objectclass: organizationalRole
cn: Manager
```

Be sure to replace `<MY-DOMAIN>` and `<COM>` with the appropriate domain components of your domain name. `<MY ORGANIZATION>` should be replaced with the name of your organization. When you cut and paste, be sure to trim any leading and trailing whitespace from the example.

```
write: dc=example,dc=com
objectclass: dcObject
objectclass: organization
o: Example Company
dc: example

dn: cn=Manager,dc=example,dc=com
```
objectclass: organizationalRole
cn: Manager

Now, you may run `ldapadd (1)` to insert these entries into your directory.

```
ldapadd -x -D "cn=Manager,dc=<MY-DOMAIN>,dc=<COM>" -W -f example.ldif
```

Be sure to replace `<MY-DOMAIN>` and `<COM>` with the appropriate domain components of your domain name. You will be prompted for the “secret” specified in `slapd.conf`. For example, for `example.com`, use:

```
ldapadd -x -D "cn=Manager,dc=example,dc=com" -W -f example.ldif
```

where `example.ldif` is the file you created above.

Additional information regarding directory creation can be found in the `Database Creation and Maintenance Tools` chapter.

**See if it works**

Now we are ready to verify the added entries are in your directory. You can use any LDAP client to do this, but our example uses the `ldapsearch (1)` tool. Remember to replace `dc=example,dc=com` with the correct values for your site:

```
ldapsearch -x -b 'dc=example,dc=com' '(objectclass=*)'
```

This command will search for and retrieve every entry in the database.

You are now ready to add more entries using `ldapadd (1)` or another LDAP client, experiment with various configuration options, backend arrangements, etc.

Note that by default, the `slapd (8)` database grants read access to everybody excepting the super-user (as specified by the `rootdn` configuration directive). It is highly recommended that you establish controls to restrict access to authorized users. Access controls are discussed in the `Access Control` chapter. You are also encouraged to read the `Security Considerations`, `Using SASL`, and `Using TLS` sections.

The following chapters provide more detailed information on making, installing, and running `slapd (8)`.

**The Big Picture**

idasdasdasd
OpenLDAP Documentation, Release 2.4.42

Distributed Schema Files

Building and Installing OpenLDAP Software

Configuring slapd

Running slapd

Access Control

Database Creation and Maintenance Tools

Backends

Schema Specification

Security Considerations

Using SASL

Using TLS

OpenLDAP Software Copyright Notices

OpenLDAP Copyright Notice

© Copyright 1998-2013 The OpenLDAP Foundation. All Rights Reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted only as authorized by the OpenLDAP Public License.

A copy of this license is available in file LICENSE in the top-level directory of the distribution or, alternatively, at http://www.openldap.org/license.html

OpenLDAP is a registered trademark of the OpenLDAP Foundation.

Individual files and/or contributed packages may be copyright by other parties and their use subject to additional restrictions.
This work is derived from the University of Michigan LDAP v3.3 distribution. Information concerning this software is available at http://www.umich.edu/~dirsvcs/ldap/ldap.html.
This work also contains materials derived from public sources.
Additional information about OpenLDAP software can be obtained at http://www.openldap.org/.

**Additional Copyright Notices**

Portions © 2001-2006 IBM Corporation.
*All rights reserved.*

Redistribution and use in source and binary forms, with or without modification, are permitted only as authorized by the [OpenLDAP Public License](http://www.openldap.org/license.html).

*All rights reserved.*

Redistribution and use in source and binary forms, with or without modification, are permitted provided that this notice is preserved. The names of the copyright holders may not be used to endorse or promote products derived from this software without their specific prior written permission. This software is provided *as is* without express or implied warranty.

**University of Michigan Copyright Notice**

Portions © 1992-1996 Regents of the University of Michigan. *All rights reserved.*

Redistribution and use in source and binary forms are permitted provided that this notice is preserved and that due credit is given to the University of Michigan at Ann Arbor. The name of the University may not be used to endorse or promote products derived from this software without specific prior written permission. This software is provided *as is* without express or implied warranty.
OpenLDAP Public License

Glossary

BDB   Berkeley DB *(Backend)*
DAP   Directory Access Protocol
DIGEST-MD5  *SASL* Digest MD5 Authentication Mechanism
DIT   Directory Information Tree
DN    Distinguished Name
DNS   Domain Name System
DSA   Directory System Agent
EXTERNAL  *SASL* External Authentication Mechanism
FAQ   Frequently Asked Questions
GSSAPI  *SASL* Kerberos V GSS-API Authentication Mechanism
IPC   Inter-Process Communication
Issue Tracking System  http://www.openldap.org/its/
HDB   Hierarchical Database *(Backend)*
JPEG  Joint Photographic Experts Group
LDAP  Lightweight Directory Access Protocol
LDAPv3  *LDAP*, version 3
LDIF  *LDAP* Data Interchange Format
OpenLDAP Foundation  http://www.openldap.org/foundation/
OpenLDAP Software  http://www.openldap.org/software/
OpenLDAP Project  http://www.openldap.org/project/
OSI   Open Systems Interconnect
RDN   Relative Distinguished Name
SASL  Simple Authentication and Security Layer
SQL   Structured Query Language
SSL   Secure Socket Layer
TLS   Transport Layer Security
University of Michigan  http://www.umich.edu/
University of Michigan LDAP Team  http://www.umich.edu/~dirsvcs/ldap/ldap.html
X.500  X.500 Directory Services
X.509  X.509 Public Key and Attribute Certificate Frameworks
Indices and tables

- genindex
- modindex
- search
Index

B
BDB, 18

D
DAP, 18
DIGEST-MD5, 18
DIT, 18
DN, 18
DNS, 18
DSA, 18

E
EXTERNAL, 18

F
FAQ, 18

G
GSSAPI, 18

H
HDB, 18

I
IPC, 18
Issue Tracking System, 18

J
JPEG, 18

L
LDAP, 18
LDAPv3, 18
LDIF, 18

O
OpenLDAP Foundation, 18
OpenLDAP Project, 18
OpenLDAP Software, 18

OSI, 18

R
RDN, 18

S
SASL, 18
SQL, 18
SSL, 18

T
The SLAPD and SLURPD Administrators Guide, 18
TLS, 18

U
University of Michigan, 18
University of Michigan LDAP Team, 18

X
X.500, 18
X.509, 18