

OpenLDAP Training Level 3

Symas Corporation

Contents

slapd.conf vs. cn=config	2
Operating System Configuration Requirements	2
Kernel Tuning	2
File System Requirements	3
Core Dump Configuration	3
Available Overlays	3
Automatic Certificate Authority	3
Auditlog	4
Chaining	4
Collective Attributes	5
Attribute Value Constraints	5
Dynamic Directory Services	5
Home Directory Provisioning	6
One Time Password	6
Proxy Bind	7
Remote Authentication	7
Server Side Sorting & Virtual List View	7
Reverse Group Membership Maintenance	7
The Proxy Cache Engine	8
Referential Integrity	8
Return Code	9
Rewrite/Remap	9
Translucent Proxy	9
Attribute Uniqueness	9
Value Sorting	10
OpenLDAP Source Code and Documentation	10
Source Code	10
OpenLDAP Documentation	10
Troubleshooting Exercises	10
Client connectivity issues	10

Slapd startup issues	10
Replication issues	10
Configuration Exercises	10
Create new attributes, objectClasses and indexes	10
Modify and testing ACLs	10
Configure a replication provider and consumer	10
Q&A	10

slapd.conf VS. cn=config

While dynamic configuration with `cn=config` has the advantage of real-time configuration changes (mostly), it lacks in readability. Static configuration with `slapd.conf` has the advantage of being readable by humans with a simple text editor.

Fortunately, it's possible to take advantage of both types of configuration. Configuration changes can be made in `slapd.conf` and then quickly convert it to `cn=config`. To do this, we use the `slaptest` command, providing the `slapd.conf` file and a directory to write the dynamic configuration:

```
# Clear the contents of the current config directory
rm -rf slapd.d/*

# Use the slaptest command to perform the conversion
slaptest -f slapd.conf -F slapd.d
```

Unfortunately, OpenLDAP doesn't provide a way to convert `cn=config` back to `slapd.conf`. However, Symas can provide a python script that is able to perform the conversion. The script requires the `python-ldap` module to be installed. The module installation requires the `python3-devel` and `openldap-devel` be installed as well:

```
dnf -install -y python3-devel openldap-devel
pip3 install python-ldap
```

The configuration source can be from an LDIF file, from the `slapcat` command or from an `ldapsearch`. For full help, run `python3 cn-config-to-slapd-conf.py --help`.

Operating System Configuration Requirements

Kernel Tuning

No specialized kernel tuning is required for OpenLDAP.

File System Requirements

OpenLDAP using the LMDB backend does not benefit from filesystem journaling as writes are always immediately synced to disk so the overhead journaling incurs is wasted. Either EXT2 or EXT4 (with journaling disabled) filesystems are the best choice.

Core Dump Configuration

By default, most operating systems are not configured to retain core dumps when applications crash. Retaining core dumps is critical for finding the cause of a slapd crash.

When enabling core dumps, it's critical that the entire core file is saved. If the size of the core dump is limited in size, any core dump larger than the limit will be truncated and be useless for inspection. To do this, the `DefaultLimitCORE` setting in `/etc/systemd/system.conf` must be enabled and be set to `infinity`.

It's also important to configure the location that the core dump is saved to a location with enough space to store the core file. Normally core files are stored in `/var/lib/systemd/coredump`, but `/var` is normally too small to store slapd core dump files. The location of core files can be set in `/etc/sysctl.conf`. The setting is `kernel.core_pattern`. See the `core(5)` man page for details.

Configuring core dumps can be tricky, so testing is required. To trigger a core dump, start slapd and execute `kill -11 $(pidof slapd)` and check the core dump location.

Available Overlays

Apart from the overlays currently in use at EC, there are many more overlays available.

Automatic Certificate Authority

`slapo-autoca(5)` Man Page

The Automatic CA overlay generates X.509 certificate/key pairs for entries in the directory. The DN of a generated certificate is identical to the DN of the entry containing it. On startup it looks for a CA certificate and key in the suffix entry of the database which it will use to sign all subsequently generated certificates. A new CA certificate and key will be generated and stored in the suffix entry if none already exists. The CA certificate is stored in the `cACertificate;binary` attribute of the suffix entry, and the private key is stored in the `cAPrivateKey;binary` attribute of the suffix entry. These attributes may be overwritten if some other CA certificate/key pair is desired for use.

Certificates for users and servers are generated on demand using a Search request returning only the `userCertificate;binary` and `userPrivateKey;binary` attributes. Any Search for anything besides exactly these two attributes is ignored by the overlay. Note that these values are stored in ASN.1 DER form in the directory so the “;binary” attribute option is mandatory.

Entries that do not belong to selected objectClasses will be ignored by the overlay. By default, entries of objectClass `person` will be treated as users, and entries of objectClass `ipHost` will be treated as servers. There are slight differences in the set of X.509V3 certificate extensions added to the certificate between users and servers.

The CA's private key is stored in a `cAPrivateKey` attribute, and user and server private keys are stored in the `userPrivateKey` attribute. The private key values are encoded in PKCS#8 format. It is essential that access to these attributes be properly secured with ACLs. Both of these attributes inherit from the `pKCS8PrivateKey` attribute, so it is sufficient to use a single ACL rule like

```
access to attrs=pKCS8PrivateKey by self ssf=128 write
```

at the beginning of the rules.

Currently there is no automated management for expiration or revocation. Obsolete certificates and keys must be manually removed by deleting an entry's `userCertificate` and `userPrivateKey` attributes.

Auditlog

`slapo-auditlog(5)` Man Page

The Audit Logging overlay can be used to record all changes on a given back-end database to a specified log file. Changes are logged as standard LDIF, with an additional comment header providing six fields of information about the change. A second comment header is added at the end of the operation to note the termination of the change.

For Add and Modify operations the identity comes from the `modifiersName` associated with the operation. This is usually the same as the requestor's identity, but may be set by other overlays to reflect other values.

Chaining

`slapo-chain` Man Page

The chain overlay provides basic chaining capability to the underlying database.

What is chaining? It indicates the capability of a DSA to follow referrals on behalf of the client, so that distributed systems are viewed as a single virtual DSA by clients that are otherwise unable to “chase” (i.e. follow) referrals by themselves.

The chaining overlay is configured in the `olcDatabase={-1}frontend,cn=config` backend.

Chaining is often used to allow replication consumers to forward modification requests to a provider.

Collective Attributes

`slapo-collect` Man Page

The collect overlay is used to provide a relatively coarse implementation of RFC 3671 collective attributes. In X.500, a collective attribute is “a user attribute whose values are the same for each member of an entry collection”.

Collective attributes are added to entries returned by a search operation when the entry is within the scope of the related ancestor. Collective attributes can only be modified when the modification affects the related ancestor.

Attribute Value Constraints

`slapo-constraint` Man Page

The constraint overlay enforces a regular expression constraint on all values of specified attributes during an LDAP modify request that contains add or modify commands. It is used to enforce a more rigorous syntax when the underlying attribute syntax is too general.

```
dn: olcOverlay={3}constraint,olcDatabase={2}mdb,cn=config
objectClass: olcConstraintConfig
olcOverlay: {3}constraint
olcConstraintAttribute: mail regex ^[[[:alnum:]]]+@mydomain.com$
olcConstraintAttribute: title uri ldap:///dc=catalog,dc=example,dc=com?title?sub?(objectClass=)
```

A specification like the above would reject any *mail* attribute which did not look like <alphanumeric string>@mydomain.com*.

It would also reject any title attribute whose values were not listed in the title attribute of any *titleCatalog* entries in the given scope.

Dynamic Directory Services

`slapo-dds` Man Page

The *dds* overlay to *slapd*(8) implements dynamic objects as per RFC2589. The name *dds* stands for Dynamic Directory Services. It allows to define dynamic objects, characterized by the *dynamicObject* objectClass.

Dynamic objects have a limited lifetime, determined by a time-to-live (TTL) that can be refreshed by means of a specific refresh extended operation. This operation allows to set the Client Refresh Period (CRP), namely the period between

refreshes that is required to preserve the dynamic object from expiration. The expiration time is computed by adding the requested TTL to the current time. When dynamic objects reach the end of their lifetime without being further refreshed, they are automatically *deleted*. There is no guarantee of immediate deletion, so clients should not count on it.

Home Directory Provisioning

slapo-homedir Man Page

The homedir overlay causes slapd(8) to notice changes involving RFC-2307bis style user-objects and make appropriate changes to the local filesystem. This can be performed on both master and replica systems, so it is possible to perform remote home directory provisioning.

One Time Password

slapo-otp Man Page

The otp overlay allows time-based one-time password, AKA “authenticator-style”, and HMAC-based one-time password authentication to be used in conjunction with a standard LDAP password for two-factor authentication.

With this overlay, users would use their password, followed with the one-time password in the password prompt to authenticate.

The password needed for a user to authenticate is calculated based on a counter (current time in case of TOTP) and a key that is referenced in the user’s LDAP entry. Since the password is based on the time or number of uses, it changes periodically. Once used, it cannot be used again so keyloggers and shoulder-surfers are thwarted. A mobile phone application, such as the Google Authenticator or YubiKey (a prover), can be used to calculate the user’s current one-time password, which is expressed as a (usually six-digit) number.

Alternatively, the value can be calculated by some other application with access to the user’s key and delivered to the user through SMS or some other channel. When prompted to authenticate, the user merely appends the code provided by the prover at the end of their password when authenticating.

This implementation complies with RFC 4226 HOTP HMAC-Based One Time Passwords and RFC 6238 TOTP Time-based One Time Passwords and includes support for the SHA-1, SHA-256, and SHA-512 HMAC algorithms.

The HMAC key used in the OTP computation is stored in the oathOTPToken entry referenced in the user’s LDAP entry and the parameters are stored in the oathOTPPParams LDAP entry referenced in the token.

Proxy Bind

`slapo-pbind` Man Page

The `pbind` overlay to `slapd(8)` forwards Simple Binds on a local database to a remote LDAP server instead of processing them locally. The remote connection is managed using an instance of the `ldap` backend.

The `pbind` overlay uses a subset of the `ldap` backend's config directives. They are described in more detail in `slapd-ldap(5)`.

Note: this overlay is built into the `ldap` backend; it is not a separate module.

Remote Authentication

`slapo-remoteauth` Man Page

The `remoteauth` overlay to `slapd(8)` provides passthrough authentication to remote directory servers, e.g. Active Directory, for LDAP simple bind operations. The local LDAP entry referenced in the bind operation is mapped to its counterpart in the remote directory. An LDAP bind operation is performed against the remote directory and results are returned based on those of the remote operation.

A `slapd` server configured with the `remoteauth` overlay handles an authentication request based on the presence of `userPassword` in the local entry. If the `userPassword` is present, authentication is performed locally, otherwise the `remoteauth` overlay performs the authentication request to the configured remote directory server.

Server Side Sorting & Virtual List View

`slapo-sssvlv` Man Page

This overlay implements the LDAP Server Side Sorting (RFC2891) control as well as the Virtual List View control. It also replaces the default implementation of the LDAP PagedResults (RFC2696) control, to ensure that it works with Sorting. The overlay can be used with any backend or globally for all backends.

Since a complete result set must be generated in memory before sorting can be performed, processing sort requests can have a large impact on the server's memory use. As such, any connection is limited to having only a limited number of sort requests active at a time. Additional limits may be configured as described below.

Reverse Group Membership Maintenance

`slapo-memberof` Man Page

In some scenarios, it may be desirable for a client to be able to determine which groups an entry is a member of, without performing an additional search. Examples of this are applications using the DIT for access control based on group authorization.

The memberof overlay updates an attribute (by default memberOf) whenever changes occur to the membership attribute (by default member) of entries of the objectclass (by default groupOfNames) configured to trigger updates.

Thus, it provides maintenance of the list of groups an entry is a member of, when usual maintenance of groups is done by modifying the members on the group entry.

The Proxy Cache Engine

slapo-pcache Man Page

LDAP servers typically hold one or more subtrees of a DIT. Replica (or shadow) servers hold shadow copies of entries held by one or more provider servers. Changes are propagated from the provider server to replica servers using LDAP Sync replication. An LDAP cache is a special type of replica which holds entries corresponding to search filters instead of subtrees.

The proxy cache extension of slapd is designed to improve the responsiveness of the ldap and meta backends. It handles a search request (query) by first determining whether it is contained in any cached search filter. Contained requests are answered from the proxy cache's local database. Other requests are passed on to the underlying ldap or meta backend and processed as usual.

Referential Integrity

slapo-refint Man Page

This overlay can be used with a backend database such as slapd-mdb(5) to maintain the cohesiveness of a schema which utilizes reference attributes.

Whenever a *modrdn* or *delete* is performed, that is, when an entry's DN is renamed or an entry is removed, the server will search the directory for references to this DN (in selected attributes: see below) and update them accordingly. If it was a *delete* operation, the reference is deleted. If it was a *modrdn* operation, then the reference is updated with the new DN.

For example, a very common administration task is to maintain group membership lists, specially when users are removed from the directory. When an user account is deleted or renamed, all groups this user is a member of have to be updated. LDAP administrators usually have scripts for that. But we can use the *refint* overlay to automate this task. In this example, if the user is removed from the directory, the overlay will take care to remove the user from all the groups he/she was a member of. No more scripting for this.

Return Code

`slapo-retcode` Man Page

This overlay is useful to test the behavior of clients when server-generated erroneous and/or unusual responses occur, for example; error codes, referrals, excessive response times and so on.

This would be classed as a debugging tool whilst developing client software or additional Overlays.

Rewrite/Remap

`slapo-rwm` Man Page

It performs basic DN/data rewrite and objectClass/attributeType mapping. Its usage is mostly intended to provide virtual views of existing data either remotely, in conjunction with the proxy backend described in *slapd-ldap(5)*, or locally, in conjunction with the relay backend described in *slapd-relay(5)*.

Translucent Proxy

`slapo-translucent` Man Page

This overlay can be used with a backend database such as *slapd-mdb(5)* to create a “translucent proxy”.

Entries retrieved from a remote LDAP server may have some or all attributes overridden, or new attributes added, by entries in the local database before being presented to the client.

A search operation is first populated with entries from the remote LDAP server, the attributes of which are then overridden with any attributes defined in the local database. Local overrides may be populated with the add, modify, and modrdn operations, the use of which is restricted to the root user of the translucent local database.

A compare operation will perform a comparison with attributes defined in the local database record (if any) before any comparison is made with data in the remote database.

Attribute Uniqueness

`slapo-unique` Man Page

This overlay can be used with a backend database such as *slapd-mdb(5)* to enforce the uniqueness of some or all attributes within a subtree.

Value Sorting

[slapo-valsort Man Page](#)

The Value Sorting overlay can be used with a backend database to sort the values of specific multi-valued attributes within a subtree. The sorting occurs whenever the attributes are returned in a search response.

OpenLDAP Source Code and Documentation

Source Code

OpenLDAP is an open source project with the source code publicly available.

[OpenLDAP git Repository](#)

OpenLDAP Documentation

[The OpenLDAP Administrators Guide](#)

[OpenLDAP Man Pages](#)

[Symas Knowledge Base](#)

[OpenLDAP Quick Start Guide](#)

Troubleshooting Exercises

Client connectivity issues

Slapd startup issues

Replication issues

Configuration Exercises

Create new attributes, objectClasses and indexes

Modify and testing ACLs

Configure a replication provider and consumer

Q&A