

**NAME**

slapd.conf – configuration file for slapd, the stand-alone LDAP daemon

**SYNOPSIS**

/etc/ldap/slapd.conf

**DESCRIPTION**

The file **/etc/ldap/slapd.conf** contains configuration information for the **slapd(8)** daemon. This configuration file is also used by the **slurpd(8)** replication daemon and by the SLAPD tools **slapacl(8)**, **slapadd(8)**, **slapauth(8)**, **slapcat(8)**, **slapdn(8)**, **slapindex(8)**, and **slaptest(8)**.

The **slapd.conf** file consists of a series of global configuration options that apply to **slapd** as a whole (including all backends), followed by zero or more database backend definitions that contain information specific to a backend instance. The configuration options are case-insensitive; their value, on a case by case basis, may be case-sensitive.

The general format of **slapd.conf** is as follows:

```
# comment - these options apply to every database
<global configuration options>
# first database definition & configuration options
database <backend 1 type>
<configuration options specific to backend 1>
# subsequent database definitions & configuration options
...
```

As many backend-specific sections as desired may be included. Global options can be overridden in a backend (for options that appear more than once, the last appearance in the **slapd.conf** file is used).

If a line begins with white space, it is considered a continuation of the previous line. Blank lines and comment lines beginning with a '#' character are ignored. Note: continuation lines are unwrapped before comment processing is applied.

Arguments on configuration lines are separated by white space. If an argument contains white space, the argument should be enclosed in double quotes. If an argument contains a double quote (") or a backslash character (\), the character should be preceded by a backslash character.

The specific configuration options available are discussed below in the Global Configuration Options, General Backend Options, and General Database Options. Backend-specific options are discussed in the **slapd-<backend>(5)** manual pages. Refer to the "OpenLDAP Administrator's Guide" for more details on the slapd configuration file.

**GLOBAL CONFIGURATION OPTIONS**

Options described in this section apply to all backends, unless specifically overridden in a backend definition. Arguments that should be replaced by actual text are shown in brackets <>.

**access to <what> [ by <who> <access> <control> ]+**

Grant access (specified by <access>) to a set of entries and/or attributes (specified by <what>) by one or more requestors (specified by <who>). If no access controls are present, the default policy allows anyone and everyone to read anything but restricts updates to rootdn. (e.g., "access to \* by \* read"). The rootdn can always read and write EVERYTHING! See **slapd.access(5)** and the "OpenLDAP's Administrator's Guide" for details.

**allow <features>**

Specify a set of features (separated by white space) to allow (default none). **bind\_v2** allows acceptance of LDAPv2 bind requests. Note that **slapd(8)** does not truly implement LDAPv2 (RFC 1777), now Historic (RFC 3494). **bind\_anon\_cred** allows anonymous bind when credentials are not empty (e.g. when DN is empty). **bind\_anon\_dn** allows unauthenticated (anonymous) bind when DN is not empty. **update\_anon** allows unauthenticated (anonymous) update operations to be processed (subject to access controls and other administrative limits).

**argsfile** <filename>

The ( absolute ) name of a file that will hold the **slapd** server's command line options if started without the debugging command line option.

**attributeoptions** [option-name]...

Define tagging attribute options or option tag/range prefixes. Options must not end with '-', prefixes must end with '-'. The 'lang-' prefix is predefined. If you use the **attributeoptions** directive, 'lang-' will no longer be defined and you must specify it explicitly if you want it defined.

An attribute description with a tagging option is a subtype of that attribute description without the option. Except for that, options defined this way have no special semantics. Prefixes defined this way work like the 'lang-' options: They define a prefix for tagging options starting with the prefix. That is, if you define the prefix 'x-foo-', you can use the option 'x-foo-bar'. Furthermore, in a search or compare, a prefix or range name (with a trailing '-') matches all options starting with that name, as well as the option with the range name sans the trailing '-'. That is, 'x-foo-bar-' matches 'x-foo-bar' and 'x-foo-bar-baz'.

RFC 4520 reserves options beginning with 'x-' for private experiments. Other options should be registered with IANA, see RFC 4520 section 3.5. OpenLDAP also has the 'binary' option built in, but this is a transfer option, not a tagging option.

**attributetype** ( <oid> [NAME <name>] [DESC <description>] [OBSOLETE] [SUP <oid>] [EQUALITY <oid>] [ORDERING <oid>] [SUBSTR <oid>] [SYNTAX <oidlen>] [SINGLE-VALUE] [COLLECTIVE] [NO-USER-MODIFICATION] [USAGE <attributeUsage>] )

Specify an attribute type using the LDAPv3 syntax defined in RFC 4512. The slapd parser extends the RFC 4512 definition by allowing string forms as well as numeric OIDs to be used for the attribute OID and attribute syntax OID. (See the **objectidentifier** description.)

**authz-policy** <policy>

Used to specify which rules to use for Proxy Authorization. Proxy authorization allows a client to authenticate to the server using one user's credentials, but specify a different identity to use for authorization and access control purposes. It essentially allows user A to login as user B, using user A's password. The **none** flag disables proxy authorization. This is the default setting. The **from** flag will use rules in the *authzFrom* attribute of the authorization DN. The **to** flag will use rules in the *authzTo* attribute of the authentication DN. The **any** flag, an alias for the deprecated value of **both**, will allow any of the above, whatever succeeds first (checked in **to**, **from** sequence. The **all** flag requires both authorizations to succeed.

The rules are mechanisms to specify which identities are allowed to perform proxy authorization. The *authzFrom* attribute in an entry specifies which other users are allowed to proxy login to this entry. The *authzTo* attribute in an entry specifies which other users this user can authorize as. Use of *authzTo* rules can be easily abused if users are allowed to write arbitrary values to this attribute. In general the *authzTo* attribute must be protected with ACLs such that only privileged users can modify it. The value of *authzFrom* and *authzTo* describes an **identity** or a set of identities; it can take five forms:

```
ldap:///<base>??[<scope>]?<filter>
dn[.<dnstyle>]:<pattern>
u[<mech>[<realm>]]:<pattern>
group[/objectClass[/attributeType]]:<pattern>
<pattern>

<dnstyle>:={exact|onelevel|children|subtree|regex}
```

The first form is a valid LDAP URI where the <host>:<port>, the <attrs> and the <extensions> portions must be absent, so that the search occurs locally on either *authzFrom* or *authzTo*. The

second form is a **DN**, with the optional style modifiers *exact*, *onelevel*, *children*, and *subtree* for exact, onelevel, children and subtree matches, which cause `<pattern>` to be normalized according to the DN normalization rules, or the special *regex* style, which causes the `<pattern>` to be treated as a POSIX ("extended") regular expression, as discussed in **regex(7)** and/or **re\_format(7)**. A pattern of `*` means any non-anonymous DN. The third form is a **SASL id**, with the optional fields `<mech>` and `<realm>` that allow to specify a SASL **mechanism**, and eventually a SASL **realm**, for those mechanisms that support one. The need to allow the specification of a mechanism is still debated, and users are strongly discouraged to rely on this possibility. The fourth form is a group specification, consisting of the keyword **group**, optionally followed by the specification of the group **objectClass** and member **attributeType**. The group with DN `<pattern>` is searched with base scope, and in case of match, the values of the member **attributeType** are searched for the asserted DN. For backwards compatibility, if no identity type is provided, i.e. only `<pattern>` is present, an *exact DN* is assumed; as a consequence, `<pattern>` is subjected to DN normalization. Since the interpretation of *authzFrom* and *authzTo* can impact security, users are strongly encouraged to explicitly set the type of identity specification that is being used. A subset of these rules can be used as third arg in the **authz-regex** statement (see below); significantly, the *URI* and the *dn.exact:<dn>* forms.

#### **authz-regex <match> <replace>**

Used by the authentication framework to convert simple user names, such as provided by SASL subsystem, to an LDAP DN used for authorization purposes. Note that the resultant DN need not refer to an existing entry to be considered valid. When an authorization request is received from the SASL subsystem, the SASL **USERNAME**, **REALM**, and **MECHANISM** are taken, when available, and combined into a name of the form

```
UID=<username>[[,CN=<realm>],CN=<mechanism>],CN=auth
```

This name is then compared against the **match** POSIX ("extended") regular expression, and if the match is successful, the name is replaced with the **replace** string. If there are wildcard strings in the **match** regular expression that are enclosed in parenthesis, e.g.

```
UID=([,]*),CN=.*
```

then the portion of the name that matched the wildcard will be stored in the numbered placeholder variable \$1. If there are other wildcard strings in parenthesis, the matching strings will be in \$2, \$3, etc. up to \$9. The placeholders can then be used in the **replace** string, e.g.

```
UID=$1,OU=Accounts,DC=example,DC=com
```

The replaced name can be either a DN, i.e. a string prefixed by "dn:", or an LDAP URI. If the latter, the server will use the URI to search its own database(s) and, if the search returns exactly one entry, the name is replaced by the DN of that entry. The LDAP URI must have no hostport, attrs, or extensions components, but the filter is mandatory, e.g.

```
ldap:///OU=Accounts,DC=example,DC=com??one?(UID=$1)
```

The protocol portion of the URI must be strictly **ldap**. Note that this search is subject to access controls. Specifically, the authentication identity must have "auth" access in the subject.

Multiple **authz-regex** options can be given in the configuration file to allow for multiple matching and replacement patterns. The matching patterns are checked in the order they appear in the file, stopping at the first successful match.

#### **concurrency <integer>**

Specify a desired level of concurrency. Provided to the underlying thread system as a hint. The default is not to provide any hint.

**conn\_max\_pending** <integer>

Specify the maximum number of pending requests for an anonymous session. If requests are submitted faster than the server can process them, they will be queued up to this limit. If the limit is exceeded, the session is closed. The default is 100.

**conn\_max\_pending\_auth** <integer>

Specify the maximum number of pending requests for an authenticated session. The default is 1000.

**defaultsearchbase** <dn>

Specify a default search base to use when client submits a non-base search request with an empty base DN. Base scoped search requests with an empty base DN are not affected.

**disallow** <features>

Specify a set of features (separated by white space) to disallow (default none). **bind\_anon** disables acceptance of anonymous bind requests. Note that this setting does not prohibit anonymous directory access (See "require authc"). **bind\_simple** disables simple (bind) authentication. **tls\_2\_anon** disables forcing session to anonymous status (see also **tls\_authc**) upon StartTLS operation receipt. **tls\_authc** disallow the StartTLS operation if authenticated (see also **tls\_2\_anon**).

**ditcontentrule** ( <oid> [NAME <name>] [DESC <description>] [OBSOLETE] [AUX <oids>] [MUST <oids>] [MAY <oids>] [NOT <oids>] )

Specify an DIT Content Rule using the LDAPv3 syntax defined in RFC 4512. The slapd parser extends the RFC 4512 definition by allowing string forms as well as numeric OIDs to be used for the attribute OID and attribute syntax OID. (See the **objectidentifier** description.)

**gentlehub** { on | off }

A SIGHUP signal will only cause a 'gentle' shutdown-attempt: **Slapd** will stop listening for new connections, but will not close the connections to the current clients. Future write operations return unwilling-to-perform, though. Slapd terminates when all clients have closed their connections (if they ever do), or – as before – if it receives a SIGTERM signal. This can be useful if you wish to terminate the server and start a new **slapd** server **with another database**, without disrupting the currently active clients. The default is off. You may wish to use **idletimeout** along with this option.

**idletimeout** <integer>

Specify the number of seconds to wait before forcibly closing an idle client connection. A idletimeout of 0 disables this feature. The default is 0.

**include** <filename>

Read additional configuration information from the given file before continuing with the next line of the current file.

**index\_substr\_if\_minlen** <integer>

Specify the minimum length for subinitial and subfinal indices. An attribute value must have at least this many characters in order to be processed by the indexing functions. The default is 2.

**index\_substr\_if\_maxlen** <integer>

Specify the maximum length for subinitial and subfinal indices. Only this many characters of an attribute value will be processed by the indexing functions; any excess characters are ignored. The default is 4.

**index\_substr\_any\_len** <integer>

Specify the length used for subany indices. An attribute value must have at least this many characters in order to be processed. Attribute values longer than this length will be processed in segments of this length. The default is 4. The subany index will also be used in subinitial and subfinal index lookups when the filter string is longer than the *index\_substr\_if\_maxlen* value.

**index\_substr\_any\_step** <integer>

Specify the steps used in subany index lookups. This value sets the offset for the segments of a filter string that are processed for a subany index lookup. The default is 2. For example, with the default values, a search using this filter "cn=\*abcdefgh\*" would generate index lookups for "abcd", "cdef", and "efgh".

**localSSF** <SSF>

Specifies the Security Strength Factor (SSF) to be given local LDAP sessions, such as those to the ldapi:// listener. For a description of SSF values, see **sasl-secprops**'s **minssf** option description. The default is 71.

**loglevel** <integer> [...]

Specify the level at which debugging statements and operation statistics should be syslogged (currently logged to the **syslogd**(8) LOG\_LOCAL4 facility). They must be considered subsystems rather than increasingly verbose log levels. Some messages with higher priority are logged regardless of the configured loglevel as soon as some logging is configured, otherwise anything is logged at all. Log levels are additive, and available levels are:

<b>1</b>	<b>(0x1 trace)</b> trace function calls
<b>2</b>	<b>(0x2 packet)</b> debug packet handling
<b>4</b>	<b>(0x4 args)</b> heavy trace debugging (function args)
<b>8</b>	<b>(0x8 conns)</b> connection management
<b>16</b>	<b>(0x10 BER)</b> print out packets sent and received
<b>32</b>	<b>(0x20 filter)</b> search filter processing
<b>64</b>	<b>(0x40 config)</b> configuration file processing
<b>128</b>	<b>(0x80 ACL)</b> access control list processing
<b>256</b>	<b>(0x100 stats)</b> stats log connections/operations/results
<b>512</b>	<b>(0x200 stats2)</b> stats log entries sent
<b>1024</b>	<b>(0x400 shell)</b> print communication with shell backends
<b>2048</b>	<b>(0x800 parse)</b> entry parsing
<b>4096</b>	<b>(0x1000 cache)</b> caching (unused)
<b>8192</b>	<b>(0x2000 index)</b> data indexing (unused)
<b>16384</b>	<b>(0x4000 sync)</b> LDAPSync replication
<b>32768</b>	<b>(0x8000 none)</b> only messages that get logged whatever log level is set

The desired log level can be input as a single integer that combines the (ORed) desired levels, both in decimal or in hexadecimal notation, as a list of integers (that are ORed internally), or as a list of the names that are shown between brackets, such that

```
loglevel 129
loglevel 0x81
loglevel 128 1
loglevel 0x80 0x1
loglevel acl trace
```

are equivalent. The keyword **any** can be used as a shortcut to enable logging at all levels (equivalent to -1). The keyword **none**, or the equivalent integer representation, causes those messages that are logged regardless of the configured loglevel to be logged. In fact, if no loglevel (or a 0 level) is defined, no logging occurs, so at least the **none** level is required to have high priority messages logged.

**moduleload** <filename>

Specify the name of a dynamically loadable module to load. The filename may be an absolute path name or a simple filename. Non-absolute names are searched for in the directories specified by the **modulepath** option. This option and the **modulepath** option are only usable if slapd was compiled with --enable-modules.

**modulepath <pathspec>**

Specify a list of directories to search for loadable modules. Typically the path is colon-separated but this depends on the operating system.

**objectclass ( <oid> [NAME <name>] [DESC <description>] [OBSOLETE] [SUP <oids>] [{ ABSTRACT | STRUCTURAL | AUXILIARY }] [MUST <oids>] [MAY <oids>] )**

Specify an objectclass using the LDAPv3 syntax defined in RFC 4512. The slapd parser extends the RFC 4512 definition by allowing string forms as well as numeric OIDs to be used for the object class OID. (See the **objectidentifier** description.) Object classes are "STRUCTURAL" by default.

**objectidentifier <name> { <oid> | <name>[:<suffix>] }**

Define a string name that equates to the given OID. The string can be used in place of the numeric OID in objectclass and attribute definitions. The name can also be used with a suffix of the form "oid.xx" in which case the value "oid.xx" will be used.

**password-hash <hash> [<hash>...]**

This option configures one or more hashes to be used in generation of user passwords stored in the userPassword attribute during processing of LDAP Password Modify Extended Operations (RFC 3062). The <hash> must be one of {SSHA}, {SHA}, {SMD5}, {MD5}, {CRYPT}, and {CLEARTEXT}. The default is {SSHA}.

{SHA} and {SSHA} use the SHA-1 algorithm (FIPS 160-1), the latter with a seed.

{MD5} and {SMD5} use the MD5 algorithm (RFC 1321), the latter with a seed.

{CRYPT} uses the **crypt(3)**.

{CLEARTEXT} indicates that the new password should be added to userPassword as clear text.

Note that this option does not alter the normal user applications handling of userPassword during LDAP Add, Modify, or other LDAP operations.

**password-crypt-salt-format <format>**

Specify the format of the salt passed to **crypt(3)** when generating {CRYPT} passwords (see **password-hash**) during processing of LDAP Password Modify Extended Operations (RFC 3062).

This string needs to be in **sprintf(3)** format and may include one (and only one) %s conversion. This conversion will be substituted with a string of random characters from [A-Za-z0-9./]. For example, "%.2s" provides a two character salt and "\$1\$%.8s" tells some versions of **crypt(3)** to use an MD5 algorithm and provides 8 random characters of salt. The default is "%s", which provides 31 characters of salt.

**pidfile <filename>**

The ( absolute ) name of a file that will hold the **slapd** server's process ID ( see **getpid(2)** ) if started without the debugging command line option.

**referral <url>**

Specify the referral to pass back when **slapd(8)** cannot find a local database to handle a request. If specified multiple times, each url is provided.

**replica-argsfile**

The ( absolute ) name of a file that will hold the **slurpd** server's command line options if started without the debugging command line option.

**replica-pidfile**

The ( absolute ) name of a file that will hold the **slurpd** server's process ID ( see **getpid(2)** ) if started without the debugging command line option.

**replicationinterval**

The number of seconds **slurpd** waits before checking the relogfile for changes.

**require <conditions>**

Specify a set of conditions (separated by white space) to require (default none). The directive may be specified globally and/or per-database; databases inherit global conditions, so per-database specifications are additive. **bind** requires bind operation prior to directory operations. **LDAPv3** requires session to be using LDAP version 3. **authc** requires authentication prior to directory operations. **SASL** requires SASL authentication prior to directory operations. **strong** requires strong authentication prior to directory operations. The strong keyword allows protected "simple" authentication as well as SASL authentication. **none** may be used to require no conditions (useful to clear out globally set conditions within a particular database); it must occur first in the list of conditions.

**reverse-lookup on | off**

Enable/disable client name unverified reverse lookup (default is **off** if compiled with `--enable-lookups`).

**rootDSE <file>**

Specify the name of an LDIF(5) file containing user defined attributes for the root DSE. These attributes are returned in addition to the attributes normally produced by slapd.

**sasl-host <fqdn>**

Used to specify the fully qualified domain name used for SASL processing.

**sasl-realm <realm>**

Specify SASL realm. Default is empty.

**sasl-secprops <properties>**

Used to specify Cyrus SASL security properties. The **none** flag (without any other properties) causes the flag properties default, "noanonymous,noplain", to be cleared. The **noplain** flag disables mechanisms susceptible to simple passive attacks. The **noactive** flag disables mechanisms susceptible to active attacks. The **nodict** flag disables mechanisms susceptible to passive dictionary attacks. The **noanonymous** flag disables mechanisms which support anonymous login. The **forwardsec** flag require forward secrecy between sessions. The **passcred** require mechanisms which pass client credentials (and allow mechanisms which can pass credentials to do so). The **minssf=<factor>** property specifies the minimum acceptable *security strength factor* as an integer approximate to effective key length used for encryption. 0 (zero) implies no protection, 1 implies integrity protection only, 56 allows DES or other weak ciphers, 112 allows triple DES and other strong ciphers, 128 allows RC4, Blowfish and other modern strong ciphers. The default is 0. The **maxssf=<factor>** property specifies the maximum acceptable *security strength factor* as an integer (see minssf description). The default is INT\_MAX. The **maxbufsize=<size>** property specifies the maximum security layer receive buffer size allowed. 0 disables security layers. The default is 65536.

**schemadn <dn>**

Specify the distinguished name for the subschema subentry that controls the entries on this server. The default is "cn=Subschema".

**security <factors>**

Specify a set of security strength factors (separated by white space) to require (see **sasl-secprops**'s **minssf** option for a description of security strength factors). The directive may be specified globally and/or per-database. **ssf=<n>** specifies the overall security strength factor. **transport=<n>** specifies the transport security strength factor. **tls=<n>** specifies the TLS security strength factor. **sasl=<n>** specifies the SASL security strength factor. **update\_ssf=<n>** specifies the overall security strength factor to require for directory updates. **update\_transport=<n>** specifies the transport security strength factor to require for directory updates. **update\_tls=<n>** specifies the TLS security strength factor to require for directory updates. **update\_sasl=<n>** specifies the SASL security strength factor to require for directory updates. **simple\_bind=<n>**

specifies the security strength factor required for *simple* username/password authentication. Note that the **transport** factor is measure of security provided by the underlying transport, e.g. `ldapi://` (and eventually IPSEC). It is not normally used.

**sizelimit** {<integer>|unlimited}

**sizelimit size** [{soft|hard|unchecked}]=<integer> [...]

Specify the maximum number of entries to return from a search operation. The default size limit is 500. Use **unlimited** to specify no limits. The second format allows a fine grain setting of the size limits. Extra args can be added on the same line. See **limits** for an explanation of the different flags.

**sockbuf\_max\_incoming** <integer>

Specify the maximum incoming LDAP PDU size for anonymous sessions. The default is 262143.

**sockbuf\_max\_incoming\_auth** <integer>

Specify the maximum incoming LDAP PDU size for authenticated sessions. The default is 4194303.

**threads** <integer>

Specify the maximum size of the primary thread pool. The default is 16; the minimum value is 2.

**timelimit** {<integer>|unlimited}

**timelimit time** [{soft|hard}]=<integer> [...]

Specify the maximum number of seconds (in real time) **slapd** will spend answering a search request. The default time limit is 3600. Use **unlimited** to specify no limits. The second format allows a fine grain setting of the time limits. Extra args can be added on the same line. See **limits** for an explanation of the different flags.

**tool-threads** <integer>

Specify the maximum number of threads to use in tool mode. This should not be greater than the number of CPUs in the system. The default is 1.

## TLS OPTIONS

If **slapd** is built with support for Transport Layer Security, there are more options you can specify.

**TLSCipherSuite** <cipher-suite-spec>

Permits configuring what ciphers will be accepted and the preference order. <cipher-suite-spec> should be a cipher specification for OpenSSL. Example:

```
TLSCipherSuite HIGH:MEDIUM:+SSLv2
```

To check what ciphers a given spec selects, use:

```
openssl ciphers -v <cipher-suite-spec>
```

**TLSCACertificateFile** <filename>

Specifies the file that contains certificates for all of the Certificate Authorities that **slapd** will recognize.

**TLSCACertificatePath** <path>

Specifies the path of a directory that contains Certificate Authority certificates in separate individual files. Usually only one of this or the **TLSCACertificateFile** is used.

**TLSCertificateFile** <filename>

Specifies the file that contains the **slapd** server certificate.

**TLSCertificateKeyFile** <filename>

Specifies the file that contains the **slapd** server private key that matches the certificate stored in the **TLSCertificateFile** file. Currently, the private key must not be protected with a password, so it is of critical importance that it is protected carefully.

**TLSDHParamFile <filename>**

This directive specifies the file that contains parameters for Diffie-Hellman ephemeral key exchange. This is required in order to use a DSA certificate on the server. If multiple sets of parameters are present in the file, all of them will be processed. Note that setting this option may also enable Anonymous Diffie-Hellman key exchanges in certain non-default cipher suites. You should append "!ADH" to your cipher suites if you have changed them from the default, otherwise no certificate exchanges or verification will be done.

**TLSRandFile <filename>**

Specifies the file to obtain random bits from when /dev/[u]random is not available. Generally set to the name of the EGD/PRNGD socket. The environment variable RANDFILE can also be used to specify the filename.

**TLSVerifyClient <level>**

Specifies what checks to perform on client certificates in an incoming TLS session, if any. The <level> can be specified as one of the following keywords:

- never** This is the default. **slapd** will not ask the client for a certificate.
- allow** The client certificate is requested. If no certificate is provided, the session proceeds normally. If a bad certificate is provided, it will be ignored and the session proceeds normally.
- try** The client certificate is requested. If no certificate is provided, the session proceeds normally. If a bad certificate is provided, the session is immediately terminated.
- demand | hard | true** These keywords are all equivalent, for compatibility reasons. The client certificate is requested. If no certificate is provided, or a bad certificate is provided, the session is immediately terminated.

Note that a valid client certificate is required in order to use the SASL EXTERNAL authentication mechanism with a TLS session. As such, a non-default **TLSVerifyClient** setting must be chosen to enable SASL EXTERNAL authentication.

**TLSCRLCheck <level>**

Specifies if the Certificate Revocation List (CRL) of the CA should be used to verify if the client certificates have not been revoked. This requires **TLSCACertificatePath** parameter to be set. <level> can be specified as one of the following keywords:

- none** No CRL checks are performed
- peer** Check the CRL of the peer certificate
- all** Check the CRL for a whole certificate chain

**GENERAL BACKEND OPTIONS**

Options in this section only apply to the configuration file section for the specified backend. They are supported by every type of backend.

**backend <databasetype>**

Mark the beginning of a backend definition. <databasetype> should be one of **bdb**, **config**, **dnssrv**, **hdb**, **ldap**, **ldbm**, **ldif**, **meta**, **monitor**, **null**, **passwd**, **perl**, **relay**, **shell**, or **sql**, depending on which backend will serve the database.

**GENERAL DATABASE OPTIONS**

Options in this section only apply to the configuration file section for the database in which they are defined. They are supported by every type of backend. Note that the **database** and at least one **suffix** option are mandatory for each database.

**database <databasetype>**

Mark the beginning of a new database instance definition. <databasetype> should be one of **bdb**, **config**, **dnssrv**, **hdb**, **ldap**, **ldbm**, **ldif**, **meta**, **monitor**, **null**, **passwd**, **perl**, **relay**, **shell**, or **sql**, depending on which backend will serve the database.

**lastmod on | off**

Controls whether **slapd** will automatically maintain the modifiersName, modifyTimestamp, creatorsName, and createTimestamp attributes for entries. It also controls the entryCSN and entryUUID attributes, which are needed by the syncrepl provider. By default, lastmod is on.

**limits <who> <limit> [<limit> [...]]**

Specify time and size limits based on who initiated an operation. The argument **who** can be any of  
anonymous | users | [dn[.<style>]=]<pattern> | group[oc[at]]=<pattern>

with

<style> ::= exact | base | onelevel | subtree | children | regex | anonymous

The term **anonymous** matches all unauthenticated clients. The term **users** matches all authenticated clients; otherwise an **exact** dn pattern is assumed unless otherwise specified by qualifying the (optional) key string **dn** with **exact** or **base** (which are synonyms), to require an exact match; with **onelevel**, to require exactly one level of depth match; with **subtree**, to allow any level of depth match, including the exact match; with **children**, to allow any level of depth match, not including the exact match; **regex** explicitly requires the (default) match based on POSIX ("extended") regular expression pattern. Finally, **anonymous** matches unbound operations; the **pattern** field is ignored. The same behavior is obtained by using the **anonymous** form of the **who** clause. The term **group**, with the optional objectClass **oc** and attributeType **at** fields, followed by **pattern**, sets the limits for any DN listed in the values of the **at** attribute (default **member**) of the **oc** group objectClass (default **groupOfNames**) whose DN exactly matches **pattern**.

The currently supported limits are **size** and **time**.

The syntax for time limits is **time[,{soft|hard}]=<integer>**, where *integer* is the number of seconds slapd will spend answering a search request. If no time limit is explicitly requested by the client, the **soft** limit is used; if the requested time limit exceeds the **hard** limit, the value of the limit is used instead. If the **hard** limit is set to the keyword *soft*, the soft limit is used in either case; if it is set to the keyword *unlimited*, no hard limit is enforced. Explicit requests for time limits smaller or equal to the **hard** limit are honored. If no limit specifier is set, the value is assigned to the **soft** limit, and the **hard** limit is set to *soft*, to preserve the original behavior.

The syntax for size limits is **size[,{soft|hard|unchecked}]=<integer>**, where *integer* is the maximum number of entries slapd will return answering a search request. If no size limit is explicitly requested by the client, the **soft** limit is used; if the requested size limit exceeds the **hard** limit, the value of the limit is used instead. If the **hard** limit is set to the keyword *soft*, the soft limit is used in either case; if it is set to the keyword *unlimited*, no hard limit is enforced. Explicit requests for size limits smaller or equal to the **hard** limit are honored. The **unchecked** specifier sets a limit on the number of candidates a search request is allowed to examine. The rationale behind it is that searches for non-properly indexed attributes may result in large sets of candidates, which must be examined by **slapd(8)** to determine whether they match the search filter or not. The **unchecked** limit provides a means to drop such operations before they are even started. If the selected candidates exceed the **unchecked** limit, the search will abort with *Unwilling to perform*. If it is set to the keyword *unlimited*, no limit is applied (the default). If it is set to *disable*, the search is not even performed; this can be used to disallow searches for a specific set of users. If no limit specifier is set, the value is assigned to the **soft** limit, and the **hard** limit is set to *soft*, to preserve the original behavior.

In case of no match, the global limits are used. The default values are the same of **sizelimit** and **timelimit**; no limit is set on **unchecked**.

If **pagedResults** control is requested, the **hard** size limit is used by default, because the request of a specific page size is considered an explicit request for a limitation on the number of entries to be returned. However, the size limit applies to the total count of entries returned within the search, and not to a single page. Additional size limits may be enforced; the syntax is **size.pr={<integer>|noEstimate|unlimited}**, where *integer* is the max page size if no explicit limit is set; the keyword *noEstimate* inhibits the server from returning an estimate of the total number of entries that might be returned (note: the current implementation does not return any estimate). The keyword *unlimited* indicates that no limit is applied to the pagedResults control page size. The syntax **size.prtotal={<integer>|unlimited|disabled}** allows to set a limit on the total number of entries that a pagedResults control allows to return. By default it is set to the **hard** limit. When set, *integer* is the max number of entries that the whole search with pagedResults control can return. Use *unlimited* to allow unlimited number of entries to be returned, e.g. to allow the use of the pagedResults control as a means to circumvent size limitations on regular searches; the keyword *disabled* disables the control, i.e. no paged results can be returned. Note that the total number of entries returned when the pagedResults control is requested cannot exceed the **hard** size limit of regular searches unless extended by the **prttotal** switch.

#### **maxderefddepth <depth>**

Specifies the maximum number of aliases to dereference when trying to resolve an entry, used to avoid infinite alias loops. The default is 1.

#### **overlay <overlay-name>**

Add the specified overlay to this database. An overlay is a piece of code that intercepts database operations in order to extend or change them. Overlays are pushed onto a stack over the database, and so they will execute in the reverse of the order in which they were configured and the database itself will receive control last of all.

#### **readonly on | off**

This option puts the database into "read-only" mode. Any attempts to modify the database will return an "unwilling to perform" error. By default, readonly is off.

**replica**        **uri=ldap[s]://<hostname>[:port]]host=<hostname>[:port]**        **[starttls=yes|critical]**  
**[suffix=<suffix> [...]] bindmethod=simple|sasl [binddn=<simple DN>] [credentials=<simple**  
**password>] [saslmec=<SASL mech>] [secprops=<properties>] [realm=<realm>]**  
**[authcId=<authentication ID>] [authzId=<authorization ID>] [attr!]=<attr list>**

Specify a replication site for this database. Refer to the "OpenLDAP Administrator's Guide" for detailed information on setting up a replicated **slapd** directory service. Zero or more **suffix** instances can be used to select the subtrees that will be replicated (defaults to all the database). **host** is deprecated in favor of the **uri** option. **uri** allows the replica LDAP server to be specified as an LDAP URI. A **bindmethod** of **simple** requires the options **binddn** and **credentials** and should only be used when adequate security services (e.g TLS or IPSEC) are in place. A **bindmethod** of **sasl** requires the option **saslmec**. Specific security properties (as with the **sasl-secprops** keyword above) for a SASL bind can be set with the **secprops** option. A non-default SASL realm can be set with the **realm** option. If the **mechanism** will use Kerberos, a kerberos instance should be given in **authcId**. An **attr list** can be given after the **attr** keyword to allow the selective replication of the listed attributes only; if the optional **!** mark is used, the list is considered exclusive, i.e. the listed attributes are not replicated. If an objectClass is listed, all the related attributes are (are not) replicated.

#### **repllogfile <filename>**

Specify the name of the replication log file to log changes to. The replication log is typically written by **slapd(8)** and read by **slurpd(8)**. See **slapd.repllog(5)** for more information. The specified file should be located in a directory with limited read/write/execute access as the replication logs may contain sensitive information.

**restrict** <oplist>

Specify a whitespace separated list of operations that are restricted. If defined inside a database specification, restrictions apply only to that database, otherwise they are global. Operations can be any of **add**, **bind**, **compare**, **delete**, **extended**[=<OID>], **modify**, **rename**, **search**, or the special pseudo-operations **read** and **write**, which respectively summarize read and write operations. The use of *restrict write* is equivalent to *readonly on* (see above). The **extended** keyword allows to indicate the OID of the specific operation to be restricted.

**rootdn** <dn>

Specify the distinguished name that is not subject to access control or administrative limit restrictions for operations on this database. This DN may or may not be associated with an entry. An empty root DN (the default) specifies no root access is to be granted. It is recommended that the rootdn only be specified when needed (such as when initially populating a database). If the rootdn is within a namingContext (suffix) of the database, a simple bind password may also be provided using the **rootpw** directive. Note that the rootdn is always needed when using *syncrepl*.

**rootpw** <password>

Specify a password (or hash of the password) for the rootdn. The password can only be set if the rootdn is within the namingContext (suffix) of the database. This option accepts all RFC 2307 userPassword formats known to the server (see **password-hash** description) as well as cleartext. **slappasswd(8)** may be used to generate a hash of a password. Cleartext and **{CRYPT}** passwords are not recommended. If empty (the default), authentication of the root DN is by other means (e.g. SASL). Use of SASL is encouraged.

**suffix** <dn suffix>

Specify the DN suffix of queries that will be passed to this backend database. Multiple suffix lines can be given and at least one is required for each database definition. If the suffix of one database is "inside" that of another, the database with the inner suffix must come first in the configuration file.

**subordinate** [advertise]

Specify that the current backend database is a subordinate of another backend database. A subordinate database may have only one suffix. This option may be used to glue multiple databases into a single namingContext. If the suffix of the current database is within the namingContext of a superior database, searches against the superior database will be propagated to the subordinate as well. All of the databases associated with a single namingContext should have identical rootdns. Behavior of other LDAP operations is unaffected by this setting. In particular, it is not possible to use *moddn* to move an entry from one subordinate to another subordinate within the namingContext.

If the optional **advertise** flag is supplied, the naming context of this database is advertised in the root DSE. The default is to hide this database context, so that only the superior context is visible.

If the slap tools **slapcat(8)**, **slapadd(8)**, or **slapindex(8)** are used on the superior database, any glued subordinates that support these tools are opened as well.

Databases that are glued together should usually be configured with the same indices (assuming they support indexing), even for attributes that only exist in some of these databases. In general, all of the glued databases should be configured as similarly as possible, since the intent is to provide the appearance of a single directory.

Note that the *subordinate* functionality is implemented internally by the *glue* overlay and as such its behavior will interact with other overlays in use. By default, the glue overlay is automatically configured as the last overlay on the superior backend. Its position on the backend can be explicitly configured by setting an **overlay glue** directive at the desired position. This explicit configuration is necessary e.g. when using the *syncprov* overlay, which needs to follow *glue* in order to work over all of the glued databases. E.g.

```

database bdb
suffix dc=example,dc=com
...
overlay glue
overlay syncprov

```

```

syncrepl          rid=<replica ID>          provider=ldap[s]://<hostname>[:port]
[type=refreshOnly|refreshAndPersist] [interval=dd:hh:mm:ss] [retry=[<retry interval> <# of
retries>]+] searchbase=<base DN> [filter=<filter str>] [scope=sub|one|base] [attrs=<attr
list>] [attronly] [sizelimit=<limit>] [timelimit=<limit>] [schemachecking=on|off]
[starttls=yes|critical] [bindmethod=simple|sasl] [binddn=<dn>] [saslmech=<mech>]
[authcid=<identity>] [authzid=<identity>] [credentials=<passwd>] [realm=<realm>]
[secprops=<properties>] [logbase=<base DN>] [logfilter=<filter str>]
[syncdata=default|accesslog|changelog]

```

Specify the current database as a replica which is kept up-to-date with the master content by establishing the current **slapd**(8) as a replication consumer site running a **syncrepl** replication engine. The replica content is kept synchronized to the master content using the LDAP Content Synchronization protocol. Refer to the "OpenLDAP Administrator's Guide" for detailed information on setting up a replicated **slapd** directory service using the **syncrepl** replication engine. **rid** identifies the current **syncrepl** directive within the replication consumer site. It is a non-negative integer having no more than three digits. **provider** specifies the replication provider site containing the master content as an LDAP URI. If <port> is not given, the standard LDAP port number (389 or 636) is used. The content of the **syncrepl** replica is defined using a search specification as its result set. The consumer **slapd** will send search requests to the provider **slapd** according to the search specification. The search specification includes **searchbase**, **scope**, **filter**, **attrs**, **attronly**, **sizelimit**, and **timelimit** parameters as in the normal search specification. The **scope** defaults to **sub**, the **filter** defaults to (**objectclass**=\*), and there is no default **searchbase**. The **attrs** list defaults to "\*,+" to return all user and operational attributes, and **attronly** is unset by default. The **sizelimit** and **timelimit** only accept "unlimited" and positive integers, and both default to "unlimited". The LDAP Content Synchronization protocol has two operation types. In the **refreshOnly** operation, the next synchronization search operation is periodically rescheduled at an interval time (specified by **interval** parameter; 1 day by default) after each synchronization operation finishes. In the **refreshAndPersist** operation, a synchronization search remains persistent in the provider **slapd**. Further updates to the master replica will generate **searchResultEntry** to the consumer **slapd** as the search responses to the persistent synchronization search. If an error occurs during replication, the consumer will attempt to reconnect according to the **retry** parameter which is a list of the <retry interval> and <# of retries> pairs. For example, **retry**="60 10 300 3" lets the consumer retry every 60 seconds for the first 10 times and then retry every 300 seconds for the next 3 times before stop retrying. The '+' in <# of retries> means indefinite number of retries until success. The schema checking can be enforced at the LDAP Sync consumer site by turning on the **schemachecking** parameter. The default is off. The **starttls** parameter specifies use of the StartTLS extended operation to establish a TLS session before Binding to the provider. If the StartTLS request fails and the **critical** argument was used, the session will be aborted. Otherwise the **syncrepl** session continues without TLS. A **bindmethod** of **simple** requires the options **binddn** and **credentials** and should only be used when adequate security services (e.g. TLS or IPSEC) are in place. A **bindmethod** of **sasl** requires the option **saslmech**. Depending on the mechanism, an authentication identity and/or credentials can be specified using **authcid** and **credentials**. The **authzid** parameter may be used to specify an authorization identity. Specific security properties (as with the **sasl-secprops** keyword above) for a SASL bind can be set with the **secprops** option. A non default SASL realm can be set with the **realm** option.

Rather than replicating whole entries, the consumer can query logs of data modifications. This mode of operation is referred to as *delta syncrepl*. In addition to the above parameters, the **logbase** and **logfilter** parameters must be set appropriately for the log that will be used. The **syncdata**

parameter must be set to either "accesslog" if the log conforms to the **slapo-accesslog(5)** log format, or "changelog" if the log conforms to the obsolete *changelog* format. If the **syncdata** parameter is omitted or set to "default" then the log parameters are ignored.

**updatedn <dn>**

This option is only applicable in a slave database updated using **slurpd(8)**. It specifies the DN permitted to update (subject to access controls) the replica (typically, this is the DN **slurpd(8)** binds to update the replica). Generally, this DN *should not* be the same as the **rootdn** used at the master.

**updateref <url>**

Specify the referral to pass back when **slapd(8)** is asked to modify a replicated local database. If specified multiple times, each url is provided.

## DATABASE-SPECIFIC OPTIONS

Each database may allow specific configuration options; they are documented separately in the backends' manual pages.

## BACKENDS

The following backends can be compiled into slapd. They are documented in the **slapd-<backend>(5)** manual pages.

- bdb** This is the recommended primary backend for a normal slapd database. It takes care to configure it properly. It uses the transactional database interface of the Sleepycat Berkeley DB (BDB) package to store data.
- config** This backend is used to manage the configuration of slapd run-time.
- dnssrv** This backend is experimental. It serves up referrals based upon SRV resource records held in the Domain Name System.
- hdb** This is a variant of the BDB backend that uses a hierarchical database layout which supports subtree renames.
- ldap** This backend acts as a proxy to forward incoming requests to another LDAP server.
- ldbm** This is an easy-to-configure but obsolete database backend. It does not offer the data durability features of the BDB and HDB backends and hence is deprecated in favor of these robust backends. LDBM uses lightweight non-transactional DB interfaces, such as those providing by GDBM or Berkeley DB, to store data.
- ldif** This database uses the filesystem to build the tree structure of the database, using plain ascii files to store data. Its usage should be limited to very simple databases, where performance is not a requirement.
- meta** This backend performs basic LDAP proxying with respect to a set of remote LDAP servers. It is an enhancement of the ldap backend.
- monitor** This backend provides information about the running status of the slapd daemon.
- null** Operations in this backend succeed but do nothing.
- passwd** This backend is provided for demonstration purposes only. It serves up user account information from the system **passwd(5)** file.
- perl** This backend embeds a **perl(1)** interpreter into slapd. It runs Perl subroutines to implement LDAP operations.
- relay** This backend is experimental. It redirects LDAP operations to another database in the same server, based on the naming context of the request. Its use requires the **rwm** overlay (see **slapo-rwm(5)** for details) to rewrite the naming context of the request. It is primarily intended to implement virtual views on databases that actually store data.

- shell** This backend executes external programs to implement LDAP operations. It is primarily intended to be used in prototypes.
- sql** This backend is experimental. It services LDAP requests from an SQL database.

## OVERLAYS

The following overlays can be compiled into slapd. They are documented in the **slapo-<overlay>(5)** manual pages.

### **accesslog**

Access Logging. This overlay can record accesses to a given backend database on another database.

### **auditlog**

Audit Logging. This overlay records changes on a given backend database to an LDIF log file. By default it is not built.

- chain** Chaining. This overlay allows automatic referral chasing when a referral would have been returned, either when configured by the server or when requested by the client.

- denyop** Deny Operation. This overlay allows selected operations to be denied, similar to the **restrict** option.

### **dyngroup**

Dynamic Group. This is a demo overlay which extends the Compare operation to detect members of a dynamic group. It has no effect on any other operations.

- dynlist** Dynamic List. This overlay allows expansion of dynamic groups and more.

### **lastmod**

Last Modification. This overlay maintains a service entry in the database with the DN, modification type, modifiersName and modifyTimestamp of the last write operation performed on that database.

- pcache** Proxycache. This overlay allows caching of LDAP search requests in a local database. It is most often used with the ldap or meta backends.

### **ppolicy**

Password Policy. This overlay provides a variety of password control mechanisms, e.g. password aging, password reuse and duplication control, mandatory password resets, etc.

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

### **retcode**

Return Code. This overlay is useful to test the behavior of clients when server-generated erroneous and/or unusual responses occur.

- rwm** Rewrite/remap. This overlay is experimental. It performs basic DN/data rewrite and objectClass/attributeType mapping.

### **syncprov**

Syncrepl Provider. This overlay implements the provider-side support for **syncrepl** replication, including persistent search functionality.

### **translucent**

Translucent Proxy. This overlay can be used with a backend database such as **slapd-bdb(5)** to create a "translucent proxy". Content of entries retrieved from a remote LDAP server can be partially overridden by the database.

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

**EXAMPLES**

Here is a short example of a configuration file:

```
include /etc/ldap/schema/core.schema
pidfile /var/slapd.pid

# Subtypes of "name" (e.g. "cn" and "ou") with the
# option ";x-hidden" can be searched for/compared,
# but are not shown. See slapd.access(5).
attributeoptions x-hidden lang-
access to attr=name;x-hidden by * =cs

# Protect passwords. See slapd.access(5).
access to attrs=userPassword by * auth
# Read access to other attributes and entries.
access to * by * read

database bdb
suffix "dc=our-domain,dc=com"
# The database directory MUST exist prior to
# running slapd AND should only be accessible
# by the slapd/tools. Mode 0700 recommended.
directory /var/lib/ldap
# Indices to maintain
index objectClass eq
index cn,sn,mail pres,eq,approx,sub

# We serve small clients that do not handle referrals,
# so handle remote lookups on their behalf.
database ldap
suffix ""
uri ldap://ldap.some-server.com/
lastmod off
```

"OpenLDAP Administrator's Guide" contains a longer annotated example of a configuration file. The original /etc/ldap/slapd.conf is another example.

**FILES**

```
/etc/ldap/slapd.conf
default slapd configuration file
```

**SEE ALSO**

**ldap(3)**, **slapd-bdb(5)**, **slapd-dnssrv(5)**, **slapd-hdb(5)**, **slapd-ldap(5)**, **slapd-ldbm(5)**, **slapd-ldif(5)**, **slapd-meta(5)**, **slapd-monitor(5)**, **slapd-null(5)**, **slapd-passwd(5)**, **slapd-perl(5)**, **slapd-relay(5)**, **slapd-shell(5)**, **slapd-sql(5)**, **slapd.access(5)**, **slapd.plugin(5)**, **slapd.replog(5)**, **slapd(8)**, **slapacl(8)**, **slapadd(8)**, **slapauth(8)**, **slapcat(8)**, **slapdn(8)**, **slapindex(8)**, **slappasswd(8)**, **slaptest(8)**, **slurpd(8)**.

Known overlays are documented in **slapo-accesslog(5)**, **slapo-auditlog(5)**, **slapo-chain(5)**, **slapo-dynlist(5)**, **slapo-lastmod(5)**, **slapo-pcache(5)**, **slapo-ppolicy(5)**, **slapo-refint(5)**, **slapo-retcode(5)**, **slapo-rwm(5)**, **slapo-syncprov(5)**, **slapo-translucent(5)**, **slapo-unique(5)**.

"OpenLDAP Administrator's Guide" (<http://www.OpenLDAP.org/doc/admin/>)

**ACKNOWLEDGEMENTS**

**OpenLDAP** is developed and maintained by The OpenLDAP Project (<http://www.openldap.org/>).  
**OpenLDAP** is derived from University of Michigan LDAP 3.3 Release.