Openldap for Windows

You can start to work with the LDAP directory service with the following credentials:

User: cn=Manager,dc=maxcrc,dc=com

Password: secret

The following client utilities are available:

Program	**Description**
ldapcompare.exe	**ldapcompare** compares one attribute value with the same attribute's value in one or more entries of a directory
ldapexop.exe	**ldapexop** issues the LDAP extended operation specified by oid or one of the special keywords whoami, cancel or refresh
ldapmodrdn.exe	**ldapmodrdn** opens a connection to an LDAP server, binds, and modifies the RDN of entries.
ldappasswd.exe	**ldappasswd** is a tool to set the password of an LDAP user. The tool uses the LDAPv3 Password Modify (RFC 3062) extended operation.
ldapRegisterChange.exe	**ldapregisterchange** uses the LDAP change notification control to receive notifications of changes to an object in Active Directory Domain Services.
ldapsearch.exe	**`ldapsearch`** is a command-line tool that opens a connection to an LDAP server, binds to it, and performs a search using a filter. The results are then displayed in the LDIF.
ldapsearchPersist.exe	**ldapsearchpersist** tool performs a persistent search on the directory and receives notification of any changes to entries within the scope of the search's result set. It uses the persistent search server control (e.g. used in eDirectory service).
ldapwhoami.exe	**ldapwhoami** opens a connection to an LDAP server, binds, and performs a whoami operation
Ldapmodify.exe	**ldapmodify** takes entry updates, defined using the LDAP Data Interchange Format (LDIF), as input and issues a corresponding LDAP request to the designated directory server. The LDIF information can be configured in a file or directly at the command-line.

The backend program can be run as an NT-service or just started by command line options. By now only local directory service operation are supported.

The following backend modules are supported:

Name	**Description**
Berkeley DB backend (BDB)	Oracle Berkeley DB (BDB) package
LDAP	The LDAP backend is not an actual database; instead it acts as a proxy to forward incoming requests to another LDAP server
LDIF	The LDIF backend is a basic storage backend that stores entries in text files in LDIF format, and exploits the filesystem to create the tree structure of the database.
SQL	The primary purpose of this backend is to PRESENT information stored in some RDBMS as an LDAP subtree without any programming (some SQL and maybe stored procedures can't be considered programming, anyway ;).
relay	The primary purpose of this backend is to map a naming context defined in a database running in the same instance into a virtual naming context, with attributeType and objectClass manipulation, if required. It requires the rwm overlay.
meta	The meta backend performs basic LDAP proxying with respect to a set of remote LDAP servers, called "targets". The information contained in these servers can be presented as belonging to a single Directory Information Tree (DIT). A basic knowledge of the functionality of the backend is recommended. This backend has been designed as an enhancement of the ldap backend. The two backends share many features (actually they also share portions of code). While the ldap backend is intended to proxy operations directed to a single server, the meta backend is mainly intended for proxying of multiple servers and possibly naming context masquerading. These features, although useful in many scenarios, may result in excessive overhead for some applications, so its use should be carefully considered.

The following overlays are linked to the slapd service:

Name	Description
seqmod	This overlay serializes concurrent attempts to modify a single entry
syncprov	This overlay supports the replication functionality
rwm	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, or locally, in conjunction with the relay backend. This overlay is extremely configurable and advanced.
pcache	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 master servers. Changes are propagated from the master server to replica (slave) servers using LDAP Sync replication. An LDAP cache is a special type of replica which holds entries corresponding to search filters instead of subtrees.
retcode	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.
translucent	This overlay can be used with a backend database 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.
dynlist	This overlay allows expansion of dynamic groups

	and lists. Instead of having the group members or list attributes hard coded, this overlay allows us to define an LDAP search whose results will make up the group or list.
memberof	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.
dds	The *dds* overlay 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.
accesslog	This overlay can record accesses to a given backend database on another database. This allows all of the activity on a given database to be reviewed using arbitrary LDAP queries, instead of just logging to local flat text files. Configuration options are available for selecting a subset of operation types to log, and to automatically prune older log records from the

	logging database. Log records are stored with audit schema to assure their readability whether viewed as LDIF or in raw form.
auditlog	The Audit Logging overlay can be used to record all changes on a given backend database to a specified log file.
ppolicy	This overlay follows the specifications contained in the draft RFC titled draft-behera-ldap-password-policy-09. While the draft itself is expired, it has been implemented in several directory servers, including slapd. Nonetheless, it is important to note that it is a draft, meaning that it is subject to change and is a work-in-progress.
refint	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. 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.
unique	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.
valsort	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.
constraint	This 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.

Sample of configuration file slapd.conf

```
# $OpenLDAP: pkg/ldap/servers/slapd/slapd.conf,v 1.23.2.8 2003/05/24 23:19:14 kurt Exp $
#
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
ucdata-path         <Installpath>/ucdata
include             <Installpath>/etc/schema/core.schema
include             <Installpath>/etc/schema/cosine.schema
include             <Installpath>/etc/schema/nis.schema
include             <Installpath>/etc/schema/inetorgperson.schema
include             <Installpath>/etc/schema/openldap.schema
include             <Installpath>/etc/schema/dyngroup.schema

# Define global ACLs to disable default read access.

# Do not enable referrals until AFTER you have a working directory
# service AND an understanding of referrals.
#referral  ldap://root.openldap.org

pidfile             <Installpath>/var/slapd.pid
argsfile    <Installpath>/var/slapd.args

# Load dynamic backend modules:
# modulepath        <Installpath>/libexec/openldap
# moduleload        back_bdb.la
# moduleload        back_ldap.la
# moduleload        back_ldbm.la
# moduleload        back_passwd.la
# moduleload        back_shell.la

# Enable TLS if port is defined for ldaps

TLSVerifyClient never
TLSCipherSuite HIGH:MEDIUM:-SSLv2
TLSCertificateFile <Installpath>/OpenLDAP-secure/certs/server.pem
TLSCertificateKeyFile <Installpath>/OpenLDAP-secure/certs/server.pem
TLSCACertificateFile <Installpath>/OpenLDAP-secure/certs/server.pem

# Sample security restrictions
#          Require integrity protection (prevent hijacking)
#          Require 112-bit (3DES or better) encryption for updates
#          Require 63-bit encryption for simple bind
# security ssf=1 update_ssf=112 simple_bind=64

# Sample access control policy:
#          Root DSE: allow anyone to read it
#          Subschema (sub)entry DSE: allow anyone to read it
#          Other DSEs:
```

```
#                       Allow self write access
#                       Allow authenticated users read access
#                       Allow anonymous users to authenticate
#           Directives needed to implement policy:
# access to dn.base="" by * read
# access to dn.base="cn=Subschema" by * read
# access to *
#           by self write
#           by users read
#           by anonymous auth
#
# if no access controls are present, the default policy is:
#           Allow read by all
#
# rootdn can always write!

#######################################################################
# bdb database definitions
#######################################################################

database bdb
suffix                  "dc=maxcrc,dc=com"
rootdn                  "cn=Manager,dc=maxcrc,dc=com"
# Cleartext passwords, especially for the rootdn, should
# be avoid.  See slappasswd(8) and slapd.conf(5) for details.
# Use of strong authentication encouraged.
rootpw                  secret
# The database directory MUST exist prior to running slapd AND
# should only be accessible by the slapd and slap tools.
# Mode 700 recommended.
directory <Installpath>/ data
dirtyread
searchstack 20
# Indices to maintain
index mail pres,eq
index objectclass pres
index default eq,sub
index sn eq,sub,subinitial
index telephonenumber
index cn
```

Server utilities:

Name	Description
slapacl.exe	**Slapacl** is used to check the behavior of the slapd in verifying access to data according to ACLs, as specified in slapd.access(5). It opens the slapd.conf(5) configuration file, reads in the access and defaultaccess directives, and then parses the attr list given on the command-line; if none is given, access to the entry pseudo-attribute is tested.
slapadd.exe	**Slapadd** is used to add entries specified in LDAP Directory Interchange Format (LDIF) to a slapd(8) database. It opens the given database determined by the database number or suffix and adds entries corresponding to the provided LDIF to the database. Databases configured as subordinate of this one are also updated, unless -g is specified. The LDIF input is read from standard input or the specified file.
slapauth.exe	**Slapauth** is used to check the behavior of the slapd in mapping identities for authentication and authorization purposes, as specified in slapd.conf(5). It opens the slapd.conf(5) configuration file, reads in the authz-policy and authz-regexp directives, and then parses the ID list given on the command-line.
slapcat.exe	**Slapcat** is used to generate an LDAP Directory Interchange Format (LDIF) output based upon the contents of a slapd(8) database. It opens the given database determined by the database number or suffix and writes the corresponding LDIF to standard output or the specified file. Databases configured as subordinate of this one are also output, unless -g is specified.
slapd.exe	**Slapd** is the stand-alone LDAP daemon. It listens for LDAP connections on any number of ports (default 389), responding to the LDAP operations it receives over these connections. slapd is typically invoked at boot time, usually out of /etc/rc.local. Upon startup, slapd normally forks and isassociates itself from the invoking tty. If configured in /etc/openldap/slapd.conf, the slapd process will print its process ID (see getpid(2)) to a .pid file, as well as the command line options during invocation to an .args file. If the -d flag is given, even with a zero argument, slapd will not fork and disassociate from the

	invoking tty.
slapdn.exe	**Slapdn** is used to check the conformance of a DN based on the schema defined in slapd(8) and that loaded. It opens the slapd.conf(5) configuration file, reads in the schema definitions, and then parses the DN list given on the command-line.
slapindex.exe	**Slapindex** is used to regenerate slapd(8) indices based upon the current contents of a database. It opens the given database determined by the database number or suffix and updates the indices for all values of all attributes of all entries. Databases configured as subordinate of this one are also re-indexed, unless -g is specified.
slappasswd.exe	**Slappasswd** is used to generate an userPassword value suitable for use with rootpw configuration directive.
slaptest.exe	**Slaptest** is used to check the conformance of the slapd.conf(5) configuration file. It opens the slapd.conf(5) configuration file, and parses it according to the general and the backend-specific rules, checking its sanity.
ucgendat.exe	helper tool building files for Berkeley database usage: ucgendat UnicodeData.txt -x CompositionExclusions.txt
hs_regex.dll	helper library
libeay32.dll	ssl library
libsasl.dll	sasl library
ssleay32.dll	ssl library