LDAP Command via the command line

This document is on about the use of LDAP via the command line instead of the GUI. The reason for this is the command lines for LDAP are more powerful and adapt especially if your adding/modifying/deleting objects in the LDAP Directory.

The commands I am referring to is ldapmodify, ldapsearch, ldapdelete, ldapadd and ldapcompare. I’m not quite sure why there are so many commands as I only need to use ldapmodify. As the command ldapmodify can search, modify, add and delete though I’m not quite sure about the ldapcompare command as I never had to use it.

Most modern Linux / Unix and Mac OS X will command with these LDAP commands with slight variations in the switches as I found out. The LDAP command and switches associated with this I will be referencing to the LDAP commands found in the Fedora Directory Server 1.x which are found in by the default directory /opt/fedora-ds/shared/bin.

First thing is to examine the valid switches for ldapmodify command, ie on my machine with the Fedora Directory Server Installed.

ashley@jhett:/opt/fedora-ds/shared/bin:542> man ldapmodify

The output is referenced in Appendix A if you want to take a look and map the switches accordingly to your ldapmodify command.

The other thing before anyone should attempt to do any LDAP modifications which include add, delete and modify any record types and objects. You need to understand the LDAP schema which relates to the inherent record type and their attributes and its relation in its usage.

Understanding the schema will be beyond this scope of this document, I probably could write about it but that in itself, would probably be a extremely large manual itself and I’m not here to teach the fundamentals.

If you have no documents on the schema on the LDAP that you are using or the record type and attributes, you can always reverse engineer it provided you have enough privileges to look at it.

The examples from here on are based on the Fedora Directory Server Installation which was detailed in the previous document. Short story is my LDAP server is a machine called jhett.csse.uwa.edu.au with Fedora Directory Server 1.02 installed in a domain called dc=csse,dc=uwa,dc=edu,dc=au.
Ie for the Fedora Directory Server 1.02, you have to logged in as the Directory Manager or the equivalent so you can view all the attributes. Which should bring you to the Fedora Management Console, now lets say we want to view the record type / attributes of a user in the LDAP directory. I would then double click on the Directory Server then proceed to the Directory -> jhett.csse.uwa.edu.au -> csse -> Right Click on People and choose Advance Properties and check “Show All Allowed Attributes” as shown in Figure 1.

![Organizational Unit for People](image)

Figure 1 – Organizational Unit for People

If you scroll you can the various attributes and corresponding values which has been set for that object.

Similarly you can do that for sub objects ie now I will examine a sub-object of people which is an actual user. If you have a user under people, Right Click on the person and choose advance properties. In this case I’ve called a user called ash which you should see something similar to Figure 2.
Now you can see that a User Information comprises of several Object Class, ie person, organizationalPerson, inetorgperson, posixAccount. Object class is very important as that will determine what attributes will be available. For example the objectclass posixAccount, will define the attributes fields for linux/unix users such as homedirectory, uidnumber, gidnumber, loginshell etc which are used in a unix/linux platform.

For those that don’t have any facilities such as looking at the objects and attributes you can use other programs just as jxplorer (www.jxplorer.org), which is an Open Source LDAP browser powered by java which should work on any platform. You can explorer the LDAP schema and look at the objects and attributes that you are interested in provided you have correct access connect privileges. Ie if you have insufficient privileges usually ie the Password attribute you will not be able to see it (Usually if you can see it it will be some sort of encrypted value ie crypt, md5, hash sha etc. It can be a plain text as well, but I’m sure people would set their policy not to store password attributes as plain text so I would hope)
Manipulating a LDAP directory comes down to basically how well you understand the schema. It is very much like manipulating a database but simpler as far as I am concerned.

Let say if I want to add a new user called Ashley, the objectclass I want our users to have is usually inetOrgPerson (Generic Information for a person), ntuser (Windows Attributes information if we using the LDAP to sync to Windows ) and posixAccount (Linux/Unix specific attributes). I would create a ashley.ldif template file ie

```
dn: uid=ashley,ou=People,dc=csse,dc=uwa,dc=edu,dc=au
changetype: add
objectclass: top
objectclass: person
objectclass: inetOrgPerson
objectclass: ntuser
objectclass: posixAccount
cn: ashley
givenName: ashley
sn: chew
ou: People
uid: ashley
gidnumber: 1000
homeDirectory: /home/staff/ashley
loginShell: /bin/zsh
uidNumber: 1234
mail: ashley@csse.uwa.edu.au
ntUserComment: Ashley Chew
ntUserDomainId: ashley
ntUserHomeDir: \uwa-csse\csse-server\staff\ashley
ntUserHomeDirDrive: H
ntUserProfile: \uwa-csse\csse-server\staff\ashley\profile.usr
userPassword: {Encryption Form}password or plain text password
```

With any LDAP command you will have to specify a object that you want to manipulate usually specified by dn. The attribute name dn stands for Distinguished Name and has to be unique in the whole LDAP directory. Typically people who use the LDAP uses cn as the Distinguish Name which usually is the person’s name. But I found that was a bad idea as we have lots of people with the same given name and surname especially at a University. So the Distinguish Name attribute which makes it unique is using the user identification (uid) that’s certainly definitely meant to be unique.

Usually the next attribute followed by the dn is the type of operation specified by the changetype that you want to perform which can be add, delete or modify. That’s why I questioned why there is the various ldap commands such as ldapadd, ldapdelete etc.
Now after specifying those two attributes the \texttt{dn} and \texttt{changetype}, usually afterwards you define the attributes you want to add, delete or modifying but again this depend on the type of record you are modifying and hence you need to understand the schema.

Then I would run the LDAP command

```
/opt/fedora-ds/shared/bin/ldapmodify -D "cn=Directory Manager" -p 636 -Z -P
/opt/fedora-ds/alias -h blackbox.csse.uwa.edu.au -a -w - -f ashley.ldif
```

This is done securely, alternatively this done via non SSL

```
/opt/fedora-ds/shared/bin/ldapmodify -D "cn=Directory Manager" -h blackbox.csse.uwa.edu.au -a -w - -f ashley.ldif
```

That’s basically it, I’ll show some more of the important \texttt{ldif} template files which will be important i.e

I.e Creating a Group and adding members

```
dn: cn=testgroup, ou=Groups,dc=csse,dc=uwa,dc=edu,dc=au
changetype: add
objectClass: top
objectClass: posixGroup
objectClass: groupofuniquenames
cn: testgroup
description: Test Group
gidNumber: 9999
uniqueMember: uid=testeruser01,ou=People,dc=csse,dc=uwa,dc=edu,dc=au
uniqueMember: uid=testeruser02,ou=People,dc=csse,dc=uwa,dc=edu,dc=au
uniqueMember: uid=testeruser03,ou=People,dc=csse,dc=uwa,dc=edu,dc=au
```

Note, this is assuming your users testeruser01, testeruser02 and testeruser03 exist and as I mentioned earlier make sure you know to reference your unique members in my case I’m using \texttt{uid} as the Distinguish Name.
Ie Modifying objects such as Changing PW for a user.

dn: uid=username,ou=People,dc=csse,dc=uwa,dc=edu,dc=au
changetype: modify
replace: userPassword
userPassword: Plain Text Password or Encryption Form

Again use the Distinguish Name, used in your schema. With the userPassword it can be plain text or the encrypted form be it hash, md5, SHA etc. Ie this is important especially if you are porting people transparently ie from a NIS / YP to LDAP transparently.

dn: uid=ashley,ou=People,dc=csse,dc=uwa,dc=edu,dc=au
changetype: modify
replace: userPassword
userPassword: {crypt}FdE2H.3fB2dAF

This is changing the password on the LDAP server with the predefined standard crypt form.

Ie Deleting objects

dn: uid=ashley,ou=People,dc=csse,dc=uwa,dc=edu,dc=au
changetype: delete

This will delete the object.

If all else fails, you can always used the Fedora Console to manage the system but as I said this is a real pain if you want to manipulate couple of hundred users at a time this is way more convenient if you can script.

If I have a bit more time, I might write a more comprehensive howto ldap via command line but this will get you limping along.
Appendix A – man ldapmodify

LDAPMODIFY(1)

NAME
ldapmodify, ldapadd - LDAP modify entry and LDAP add entry tools

SYNOPSIS
   [-X authzid] [-Y mech] [-Z[Z]] [-f file]

   [-X authzid] [-Y mech] [-Z[Z]] [-f file]

DESCRIPTION
ldapmodify is a shell-accessible interface to the ldap_modify(3) and ldap_add(3)
library calls. ldapadd is implemented
as a hard link to the ldapmodify tool. When invoked as ldapadd the -a (add new entry) flag is turned on automatically.

ldapmodify opens a connection to an LDAP server, binds, and modifies or adds entries. The entry information is read
from standard input or from file through the use of the -f option.

OPTIONS
-a Add new entries. The default for ldapmodify is to modify existing entries. If invoked as ldapadd, this flag is
   always set.

-c Continuous operation mode. Errors are reported, but ldapmodify will continue with modifications. The default
   is to exit after reporting an error.

-S file
   Add or change records which where skipped due to an error are written to file and the error message returned by
   the server is added as a comment. Most useful in conjunction with -c.
-n Show what would be done, but donât actually modify entries. Useful for debugging in conjunction with -v.

-v Use verbose mode, with many diagnostics written to standard output.

-k Use Kerberos IV authentication instead of simple authentication. It is assumed that you already have a valid ticket granting ticket. You must compile with Kerberos support for this option to have any effect.

-K Same as -k, but only does step 1 of the Kerberos IV bind. This is useful when connecting to a slapd and there is no x500dsa.hostname principal registered with your Kerberos Domain Controller(s).

-F Force application of all changes regardless of the contents of input lines that begin with replica: (by default, replica: lines are compared against the LDAP server host and port in use to decide if a replog record should actually be applied).

-M[M] Enable manage DSA IT control. -MM makes control critical.

d debuglevel
   Set the LDAP debugging level to debuglevel. ldapmodify must be compiled with LDAP_DEBUG defined for this option to have any effect.

-f file
   Read the entry modification information from file instead of from standard input.

-x Use simple authentication instead of SASL.

-D binddn
   Use the Distinguished Name binddn to bind to the LDAP directory.

-W Prompt for simple authentication. This is used instead of specifying the password on the command line.

-w passwd
   Use passwd as the password for simple authentication.

-y passwdfile
   Use complete contents of passwdfile as the password for simple authentication.
-H ldapuri
   Specify URI(s) referring to the ldap server(s).

-h ldaphost
   Specify an alternate host on which the ldap server is running. Deprecated in favor of -H.

-p ldapport
   Specify an alternate TCP port where the ldap server is listening. Deprecated in favor of -H.

-P 2|3 Specify the LDAP protocol version to use.

-O security-properties
   Specify SASL security properties.

-l Enable SASL Interactive mode. Always prompt. Default is to prompt only as needed.

-Q Enable SASL Quiet mode. Never prompt.

-U authcid
   Specify the authentication ID for SASL bind. The form of the ID depends on the actual SASL mechanism used.

-R realm
   Specify the realm of authentication ID for SASL bind. The form of the realm depends on the actual SASL mechanism used.

-X authzid
   Specify the requested authorization ID for SASL bind. authzid must be one of the following formats: dn:<distinguished name> or u:<username>

-Y mech
   Specify the SASL mechanism to be used for authentication. If it's not specified, the program will choose the best mechanism the server knows.

-Z[Z] Issue StartTLS (Transport Layer Security) extended operation. If you use -ZZ, the command will require the operation to be successful.

INPUT FORMAT
   The contents of file (or standard input if no -f flag is given on the command line) should conform to the format
defined in slapd.replog(5), with the exceptions noted below.

Lines that begin with "replica:"

replog record should be applied. Any other lines that precede the "dn:" line are ignored. The -F flag can be used to
force ldapmodify to apply all of the replog changes, regardless of the presence or absence of any "replica:" lines.

If no "changetype:" line is present, the default is "add" if the -a flag is set (or if the
program was invoked as ldapad"

If changetype is "modify" and no "add:", "replace:", or "delete:" lines appear, the
default is "replace" for ldapmod-
ify(1) and "add" for ldapadd(1).

Note that the above exceptions to the slapd.replog(5) format allow ldif(5) entries to
be used as input to ldapmodify or ldapadd.