

**Deploying LDAP-UX with
Novell eDirectory,
Oracle Internet Directory
&
OpenLDAP**

Version 1.0

**Doug Lamoureux
douglas.lamoureux@hp.com
Advanced Technology Center
Systems Networking Solutions Lab
Hewlett-Packard Company**

E0300

©Copyright 2003 Hewlett-Packard Development Company, L.P.

Legal Notices

The information in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty. A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

Restricted Rights Legend. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

Hewlett-Packard Company
19420 Homestead Road
Cupertino, California 95014 U.S.A.

Use of this manual and flexible disk(s) or tape cartridge(s) supplied for this pack is restricted to this product only. Additional copies of the programs may be made for security and back-up purposes only. Resale of the programs in their present form or with alterations is expressly prohibited.

Copyright Notices

©Copyright 1983-2001 Hewlett-Packard Company, all rights reserved.

©Copyright 2003 Hewlett-Packard Development Company, L.P.

Reproduction, adaptation, or translation of this document without prior written permission is prohibited, except as allowed under the copyright laws.

Contents

Legal Notices.....	1-2
Contents.....	1-3
1 Audience.....	1-5
2 Abstract.....	2-6
3 LDAP-UX Uncovered.....	3-7
3.1 Prepare Your Directory.....	3-7
3.2 Running the LDAP-UX setup tool.....	3-7
3.3 Configuration Profile.....	3-8
3.4 Proxy Users.....	3-9
3.5 POSIX Attribute (RFC 2307).....	3-10
3.6 Access Control.....	3-10
3.6.1 Configuration Profile.....	3-11
3.6.2 User (posixAccount) Objects.....	3-11
3.6.3 Group (posixGroup) Objects.....	3-11
4 Deploying LDAP-UX with Novell eDirectory.....	4-12
4.1 Preparing eDirectory for LDAP-UX Integration.....	4-12
4.1.1 Enabling Clear Text Passwords.....	4-12
4.1.2 Configure a Proxy User.....	4-13
4.1.3 Extend your eDirectory Schema to support RFC2307 (NIS/POSIX Attributes) .	4-13
4.1.4 Give users permission to change their own password.....	4-13
4.2 Profile Schema.....	4-14
4.3 Proxy Users.....	4-15
4.3.1 Deploying LDAP-UX with a Proxy User.....	4-15
4.3.2 Deploying LDAP-UX without a Proxy User.....	4-16
4.4 POSIX Attributes (RFC 2307).....	4-19
4.5 Access Control.....	4-21
4.6 Password Management.....	4-27
4.7 Migrating and Managing Users and Groups.....	4-28
4.7.1 Migrating UNIX Users and Groups into eDirectory.....	4-28
4.7.2 Managing UNIX Users and Groups in eDirectory.....	4-30
4.8 Helpful Links.....	4-33
5 Oracle Internet Directory.....	5-35
5.1 Preparing Oracle Internet Directory for LDAP-UX Integration.....	5-35
5.1.1 Updating the LDAP Operational Attribute supportedLDAPVersion.....	5-35
5.1.2 Configure a Proxy User.....	5-38
5.1.3 Extend the eDirectory Schema for RFC2307 (NIS/POSIX Attributes).....	5-38
5.1.4 Creating a Password Policy.....	5-38
5.2 Profile Schema.....	5-39
5.3 Proxy Users.....	5-43
5.3.1 Creating a Proxy User.....	5-43
5.4 POSIX Attributes (RFC 2307).....	5-44
5.4.1 Indexing the gidnumber and uidnumber attributes using ldapmodify.....	5-45
5.4.2 Indexing the gidnumber and uidnumber attributes after they have been used .	5-45
5.5 Access Control.....	5-47
5.5.1 Configuration Profile.....	5-48
5.5.2 Posix Attributes.....	5-53
5.6 Password Management.....	5-58
5.6.1 Oracle Internet Directory Password Policy.....	5-58

5.7	Migrating and Managing Users and Groups.....	5-62
5.8	Gotcha's.....	5-63
5.9	Helpful Links.....	5-64
6	OpenLDAP	6-65
6.1	Preparing OpenLDAP for LDAP-UX Integration	6-65
6.1.1	Adding the DUAConfig Profile Schema definition.....	6-65
6.1.2	Extend the OpenLDAP Schema for RFC2307 (NIS/POSIX Attributes)	6-66
6.1.3	Configure a Proxy Server (<i>Optional</i>).....	6-66
6.1.4	Access Control (ACL's)	6-67
6.2	Profile Schema.....	6-68
6.3	Proxy User	6-69
6.4	Configuring LDAP-UX client	6-70
6.5	Enabling ldapclientd.....	6-72
6.6	Enabling LDAP-UX.....	6-73
6.7	POSIX Attributes (RFC 2307).....	6-75
6.8	Access Control	6-75
6.8.1	Configuration Profile.....	6-75
6.8.2	Posix Attributes.....	6-75
6.9	Password Management.....	6-77
6.10	Migrating and Managing Users and Groups	6-77
7	Scripts.....	7-78
7.1	posixAccount_acl_check.....	7-78
7.2	posixGroup_acl_check.....	7-80
	Appendix A (<i>eDirectory RFC2307 Schema Files</i>).....	7-83
	Appendix B (<i>Oracle Internet Directory RFC2307 LDIF File</i>).....	7-93
	Appendix C (<i>DUAConfig Profile Definition for OpenLDAP</i>).....	7-98
	Appendix D (<i>Sample DUAConfig Profile Entry</i>).....	7-101

1 Audience

This document is intended for HP-UX and Directory Administrators who are familiar with LDAP concepts, LDAP based directory servers and the LDAP-UX product. The reader is expected to have very good knowledge of the Directory Server in which LDAP-UX will be deployed with. While some administration and configuration of LDAP-UX will be discussed it is important for the reader to review existing LDAP-UX documentation prior to deploying LDAP-UX.

Recommending reading (available from <http://docs.hp.com/hpux/internet/>):

- **Preparing your LDAP Directory for HP-UX Integration**
- **Installing and Administering LDAP-UX Client Services**
- **LDAP-UX Integration Release Notes**

2 Abstract

The HP LDAP-UX product, available on HP-UX 11.0 and later, allows administrators to store and authenticate users in LDAP v3 compliant Directory Servers. Along with storing users and groups it is possible to store other NIS Maps in the directory server, however this document focuses on storing and authenticating users (posixAccount) and groups (posixGroup).

By deploying LDAP-UX, along with an LDAP v3 compliant Directory Server, NIS "like" data can be accessed throughout the Enterprise using standards based protocols. Hewlett-Packard has extensively tested the LDAP-UX product with Netscape/iPlanet Directory Servers and Microsoft's Active Directory, however only limited testing has been done with other LDAP v3 compliant Directories. The information provided in this document is intended to describe how the LDAP-UX product can be deployed with other popular LDAP v3 compliant Directories. This document should be used as a reference, and does not imply that LDAP-UX will function correctly with these Directory Servers. While some testing has been performed an exhaustive test suite has not been run against these Directory Servers; therefore Hewlett-Packard does not officially support LDAP-UX with these Directory Servers. With the information provided in this document it is expected that readers will be able to deploy LDAP-UX with other Directory Servers not mentioned here.

3 LDAP-UX Uncovered

While it's not absolutely necessary to understand the internals of the LDAP-UX product it will be helpful to understand the basic product flow of LDAP-UX configuration when troubleshooting problems. This chapter is broken down into several sections that describe various components of the LDAP-UX configuration and product architecture. In the chapters that follow each Directory Server will be discussed specifically, where possible the section headings in these subsequent chapters will be the same as those in this chapter.

3.1 Prepare Your Directory

There will always be something that needs to be done to the Directory Server prior to configuring LDAP-UX. It's very important that any steps listed in this section are done prior to configuring LDAP-UX otherwise problems will arise. For a detail discussion on preparing your Directory for LDAP-UX see the White paper "Preparing your LDAP Directory for HP-UX Integration" at:

<http://docs.hp.com/hpux/internet/#LDAP-UX%20Integration>

General steps to take for any Directory Server:

- Obtain the Directory Manager's (Superuser) distinguished name and password. Configuring LDAP-UX for the first time requires that the Directory Server's schema be extended.
- Plan and create your directory tree, this may already be done when using an established Directory.
- Create a proxy user, if you plan on using one.
- Extend the Directory Server's Schema to support RFC2307 (NIS/POSIX) attributes. While it's not necessary to do this prior to configuring LDAP-UX it will need to be done before LDAP-UX is enabled for use. This will likely be a manual step outside of the normal LDAP-UX configuration, so now may be a good time to take care of this.

3.2 Running the LDAP-UX setup tool

The LDAP-UX setup tool will handle most of the configuration for your HP-UX system if it can be run against the Directory Server. There are some Directory Servers that don't support online schema updates that the setup tool will not work with. For these Directory Servers it's valuable to understand what the setup tool does since those tasks will need to be done manually. The setup tool:

1. Determines if the Directory Server supports LDAP Version 3
2. Examines the Directory Servers schema to see if the DUAConfigProfile objectclass exist. If not setup will extend the Directory Servers schema with the DUAConfigProfile objectclass and supporting attributes
3. Creates the Configuration Profile based on information entered by the user and adds it to the Directory Server
4. Configures the ldapux_client.conf file with:
 - a. Identity of the Directory Server where the configuration profile is stored
 - b. The Distinguished Name of the configuration profile entry
5. Enables and starts the LDAP client daemon (ldapclientd)

6. Configures (does not create) the proxy user's DN and credentials (password) to be used by the local LDAP-UX client (optional). NOTE: The proxy user must already exist in the Directory (See Preparing your Directory)

The LDAP-UX product stores its shared configuration information in the configuration profile entry in the Directory. By storing this configuration information in the Directory multiple LDAP-UX clients are able to use the same configuration data without having to update each client with new configuration data manually. With configuration information stored in the Directory two issues need to be addressed:

1. How does the LDAP-UX client know where to find its configuration?
2. How does the LDAP-UX client receive configuration updates?

As mentioned above, the setup tool configured the `/etc/opt/ldapux/ldapux_client.conf` file with the identity of the Directory Server that holds the configuration and the location of the configuration within the Directory. Consider this information as a *primer (bootstrap)* to retrieve the configuration from the Directory.

Now that we know where to find the configuration profile there are 2 possible ways to retrieve it from the Directory. The first option to retrieve the configuration profile is manually using the `/opt/ldapux/config/get_profile_entry` tool. This tool will read the `ldapux_client.conf` file to find where the configuration profile is located and retrieve a new copy. The second option is to let the `ldapclientd` automatically pull the configuration profile from the Directory Server. One of the configuration profile attributes (`profilettl`) is used to determine how long the client should wait before re-reading the configuration profile entry. Along with updating the local copy of the configuration profile the `ldapclientd` also provides other features such as entry caching. Please note that prior to LDAP-UX 3.2 `ldapclientd` is **not** required to run for LDAP-UX to function properly, however `ldapclientd` is required for certain features such as X.500 group membership, automatic profile downloads, and others. See the `ldapclientd` man page for additional details.

3.3 Configuration Profile

If the setup tool is used to configure the LDAP-UX client there is no need to modify any of the LDAP-UX configuration with the exception of the `pam.conf` and `nsswitch.conf` files.

If the setup tool is unable to be used the following manual steps must be taken to configure the LDAP-UX client:

1. The following lines must be added to the `ldapux_client.conf` file:


```
Service: NSS
LDAP_HOSTPORT="<Directory Server IP>:<Port>"
PROFILE_ENTRY_DN="<Configuration Profile Distinguished Name>"
PROGRAM="/opt/ldapux/config/create_profile_cache"
```
2. Extend the Directory Servers Schema to support the `DUACfgProfile` objectclass.
3. Create the configuration profile (upload to the Directory if necessary.)
4. Download the configuration profile using the `get_profile_entry` tool.
5. Manually configure the proxy user (if applicable.)
6. Start, and enable automatic start after reboot, of `ldapclientd` (if applicable.)

See specific chapters for sample LDIF files for #2 and #3

3.4 Proxy Users

Some Directory Servers, Windows 2000 Active Directory (by default) for example, require that all LDAP access to the Directory be done over an authenticated connection. It may also be desirable to set access control on the Directory such that only UNIX users have access to the UNIX specific information.

The LDAP-UX product can be configured such that all access to the Directory from the LDAP-UX client is done using an authenticated connection. For LDAP-UX to authenticate itself against the Directory the configuration profile must be modified to direct LDAP-UX to use a “proxy” user when authenticating to the Directory (as opposed to anonymous authentication). This can be done using the setup tool (answer no when prompted to accept the remaining defaults) or modifying the credentialLevel attribute of the configuration profile.

Modifying the configuration profile only directs LDAP-UX to use a proxy user; it does not tell each LDAP-UX client the name (DN) or password of the proxy user. The proxy user name and password are stored locally on each LDAP-UX client system and must be configured on every client. Again, this can be done while running the setup tool (only for the LDAP-UX client setup is being run on). The local proxy user can also be configured using the /opt/ldapux/config/ldap_proxy_config tool:

```
# /opt/ldapux/config/ldap_proxy_config -d <Proxy User DN> -c <Proxy User Password>
```

The proxy user configuration can be verified using:

```
# /opt/ldapux/config/ldap_proxy_config -v
```

Neither the setup tool nor ldap_proxy_config tool creates the proxy user in the Directory. The proxy user should be created in the Directory prior to running ldap_proxy_config or the setup tool.

If the Directory Server allows anonymous access to Directory information it is not necessary to configure proxy access for LDAP-UX.

3.5 POSIX Attribute (RFC 2307)

HP-UX, as well as most versions of UNIX, uses a number of “databases” to store information required by users and applications running on the system. Historically these “databases” were files stored on the local system, `/etc/passwd` and `/etc/hosts` for example, or in a Network repository like NIS. Applications that use standard API’s like `getpwent()` expect certain pieces of information regardless of where the information is stored. RFC 2307 (<http://www.ietf.org/rfc/rfc2307.txt>) defines how these NIS map entries are defined as X.500 entries.

With LDAP-UX’s flexible configuration it is not necessary for the Directory Server to support the exact definition of RFC 2307, although it does make configuration much easier. If the Directory Server does not support RFC 2307 LDAP-UX can be configured to “map” the standard RFC 2307 attribute to a different attribute. For example the `posixAccount` objectclass defines the user’s UID number as `uidNumber`. LDAP-UX can be configured to use a different attribute, `employeeNumber` for example, when processing a user’s entry. These attribute mappings are set in the configuration profile:

```
attributemap: passwd:uidNumber=employeeNumber
```

While running the LDAP-UX setup tool it is possible to re-map the RFC 2307 attributes, as well as modifying the configuration profile after it has been created. Generally speaking attribute mapping should be done between attributes that use the same syntax and serve a similar purpose. For example mapping the `uidNumber` attribute to the `streetNumber` attribute may not be a good idea. As with most “*rules*” there are a few exceptions:

1. The “`memberUid`” attribute, a member of the `posixGroup` objectclass, can also be mapped to “`member`” and “`memberUid`” syntaxes (which are of DN and DN+UniqueID syntax. LDAP-UX will translate from the DN syntaxes to the `memberUid` syntax (DN → UNIX-User-ID).
2. The `gecos` attribute, a member of the `posixAccount` objectclass, can be mapped to multiple attributes (such as `cn`, `telephone`, `building`.)
3. The “`cn`” attribute as part of the `ipService`, `ipProtocol` and `ipHost` objectclasses can be mapped to two attributes, the first representing the canonical name of that service, while the other attribute represents alias names.

See the “LDAP-UX Client Services Object Classes” section of the *Installing and Administering LDAP-UX Client Services* manual for more information on attribute mapping.

3.6 Access Control

Now that data is being stored in a new repository it’s important that the proper level of access control is placed on that information. Since this document focuses on using the Directory for user and group storage it will not discuss access control on other NIS maps that may be served by the Directory Server. For an in-depth discussion on security with LDAP-UX please see the White paper “Preparing your LDAP Directory for HP-UX Integration” at: <http://docs.hp.com/hpux/internet/#LDAP-UX%20Integration>

3.6.1 Configuration Profile

The configuration profile stored in the Directory must not be modifiable by an average user. If any user can modify the configuration profile it's possible for that user to make changes to the configuration that could allow them to gain elevated privileges. For example the profile could be modified to point the LDAP-UX client to a Directory Server that is under the control of the modifier. The modifier could create a user in his Directory Server that has root level privileges (`uidnumber=0`) allowing him to gain root access to the LDAP-UX system.

When the configuration profile entry is created in the Directory it will have a default level of access control set. This ACL is dependant on the Directory Server, not LDAP-UX, and should be verified before deploying this configuration into production environments. As a start the administrator should attempt to modify the configuration profile as a "regular" non-privileged user, as well as an anonymous (un-authenticated) user.

3.6.2 User (`posixAccount`) Objects

It is OK for some user information, like Gecos, to be modifiable by the owner, however it's very important that some information like `uidNumber` not be modifiable. If the user can modify their own `uidNumber` it's possible they could gain elevated privileges (setting `uidNumber` to 0 for example). Again, it's up to the Directory Server to apply default access control to users entries, its up to the administrator to ensure that ACL's are set correctly for their environment. See the section titled **Scripts** for a sample `posixAccount_acl_check` script to help test access control.

3.6.3 Group (`posixGroup`) Objects

Normally there is nothing within a UNIX/POSIX group object that a non-privileged user should be allowed to modify. One possible exception would be to allow self-enrolment to a group, though this may not be a practical use of group membership within a UNIX environment. The 3 attributes to verify proper access control against are: `userPassword`, `memberUid` and `gidNumber`. A sample ACL checking script, `posixGroup_acl_check`, is available in the Scripts section below.

4 Deploying LDAP-UX with Novell eDirectory

Novell eDirectory is available on several platforms (Windows 2000, Solaris, Tru64, Netware and HP-UX). The following information was obtained while testing with eDirectory 8.6.2 on Solaris 8. There have been successful deployments of LDAP-UX with eDirectory on Netware and Windows 2000 as well. The following pages address configuring eDirectory to store users and groups, while it is possible to store other NIS Maps in eDirectory this document does not discuss it.

To successfully integrate LDAP-UX the following actions must be taken, discussed in further detail below, in order:

1. Prepare eDirectory for LDAP-UX Integration
 - a. Clear text password
 - b. Create a proxy user
 - c. Extend schema to support RFC2307
 - d. ACL's
2. Run the LDAP-UX setup tool
3. Configure `/etc/pam.conf` & `/etc/nsswitch.conf` (see `pam.ldap` and `nsswitch.ldap` for examples)
4. Test the LDAP-UX configuration (you'll need at least 1 POSIX user/group configured in eDirectory)
5. Migrate and/or configure UNIX users and groups

4.1 Preparing eDirectory for LDAP-UX Integration

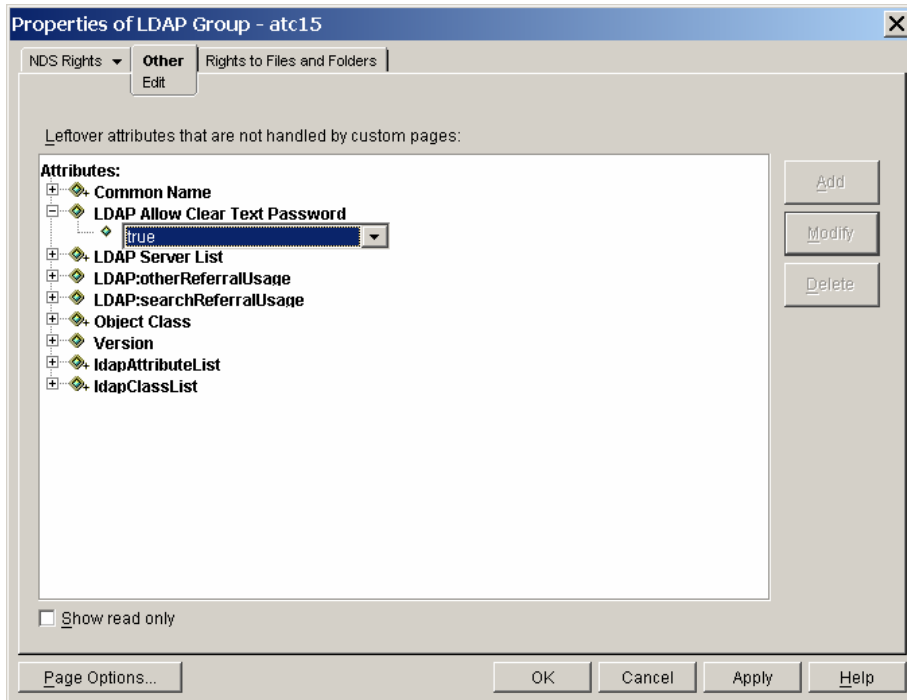
Before configuring LDAP-UX on your HP-UX system there are a few issues that need to be addressed within eDirectory.

4.1.1 Enabling Clear Text Passwords

Prior to LDAP-UX version 3.2 only simple (passwords transmitted in clear text) and Digest-MD5 authentication is supported. By default eDirectory does not allow LDAP clients to authenticate when sending a password across the network in clear text. You must enable "Clear Text Password Authentication" in eDirectory by modifying the LDAP Group object.

NOTE: This does not mean that passwords will be stored in eDirectory in clear text, only that eDirectory will allow LDAP clients to authenticate by sending clear text passwords across the network.

Using ConsoleOne set the value of “LDAP Allow Clear Text Password” to true in order to allow clear text passwords (sent over the network):



4.1.2 Configure a Proxy User

In a default eDirectory installation/configuration a proxy user is required for LDAP-UX to function properly with eDirectory. It is possible to modify the Access Control on eDirectory to allow LDAP-UX to work without using a proxy user. For more details on configuring proxy user access see the section title proxy users below.

4.1.3 Extend the eDirectory Schema to support RFC2307 (NIS/POSIX Attributes)

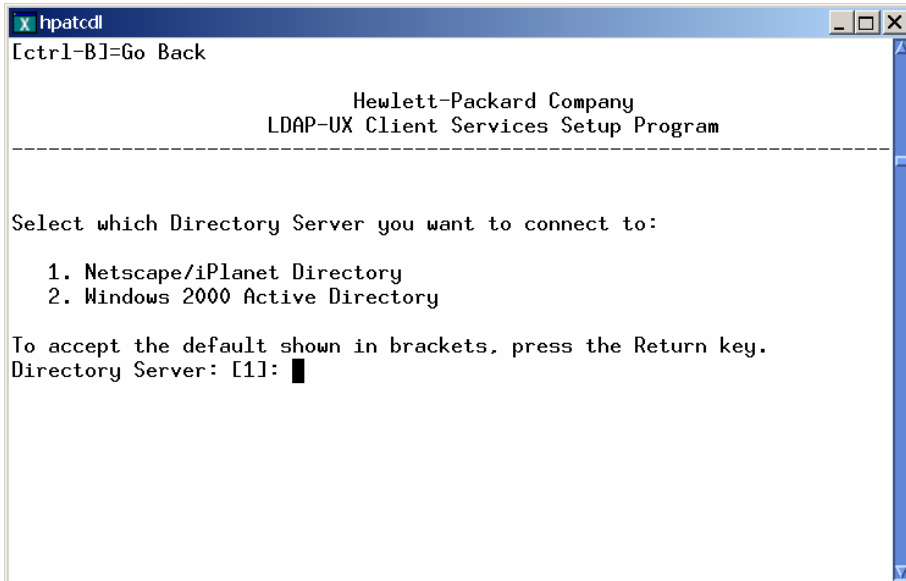
While it's not necessary to extend the eDirectory schema prior to configuring LDAP-UX it must be done prior to adding POSIX/UNIX users (or attributes to existing eDirectory users) to the Directory. See the section below “*Posix Attributes (RFC2307)*” for more information.

4.1.4 Give users permission to change their own password

By default a UNIX user will not be able to change their password from the HP-UX system. To enable password changing from the HP-UX system a user must have write access to the PasswordManagement attribute

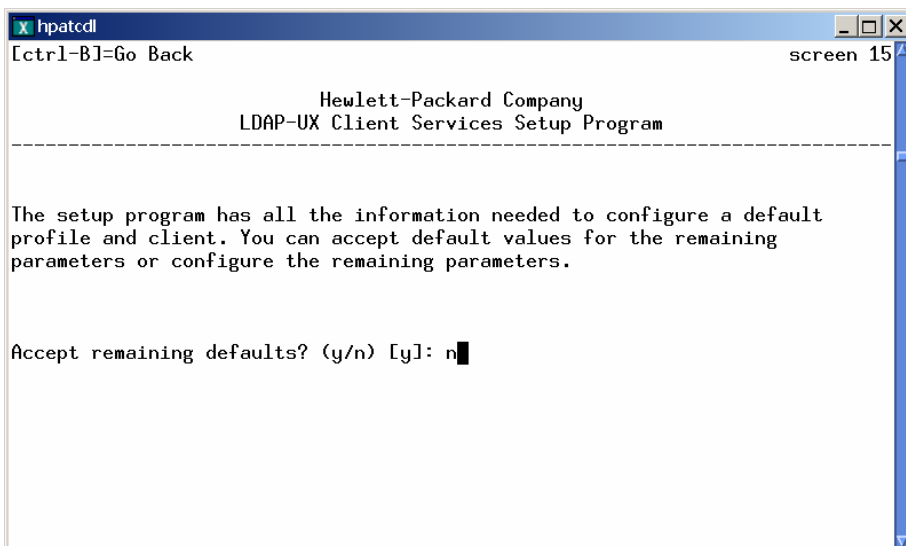
4.2 Profile Schema

Beginning with LDAP-UX 3.0 the setup tool can be used to successfully extend the eDirectory Schema to support the LDAP-UX Profile Schema (DUAConfigProfile). When running the setup tool you will be asked to select the Directory Server to connect to:



Select 1 (Netscape/iPlanet Directory).

Follow the instructions as if you were configuring LDAP-UX for the Netscape/iPlanet Directory Server, do not select Digest-MD5 authentication; it is not supported with eDirectory. In order to configure Proxy Client binding you must answer NO when asked to "Accept remaining defaults?":



4.3 Proxy Users

In the default configuration/installation of eDirectory 8.6.2 on Solaris 8 it is necessary to configure a proxy user in order for LDAP-UX to function correctly. However, it is possible to modify access control on the eDirectory to allow LDAP-UX to work without configuring a proxy user.

NOTE: Using the default configuration/installation of the RFC2307 Schema extensions for eDirectory on Solaris anonymous access was granted to the POSIX attributes of user and group objects. Therefore no access control modifications on a user or group object are required, regardless of whether or not a proxy user is used

4.3.1 Deploying LDAP-UX with a Proxy User

The default access control of eDirectory does not allow an anonymous user/bind to download the LDAP-UX Configuration Profile:

```
# ldapsearch -h ldap.hp.com -b "o=hp" cn=ldapuxprofile
```

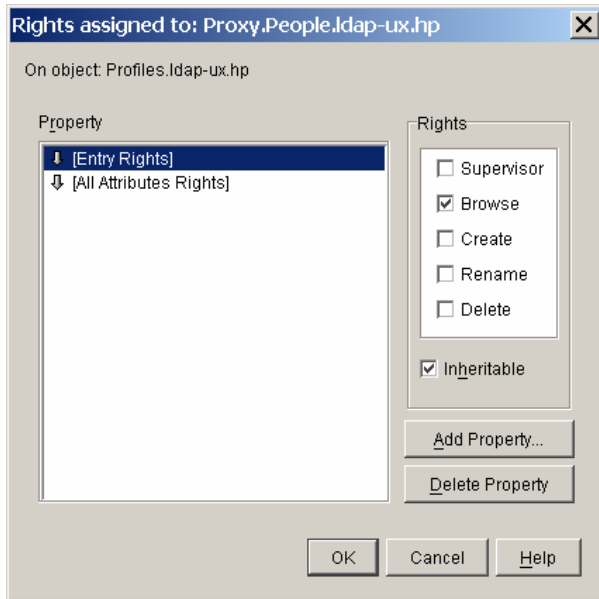
Additionally by default a non-privileged/non-admin user cannot view the profile, even when properly authenticating to the Directory:

```
# ldapsearch -h ldap.hp.com -D "cn=proxy,ou=People,ou=ldap-ux,o=hp" -w PaSSWord \
-b "o=hp" cn=ldapuxprofile
```

However the Directory Administrator is able to read the profile:

```
# ldapsearch -h ldap.hp.com -D "cn=admin,o=hp" -w pAsSWord -b "o=hp" cn=ldapuxprofile
dn: cn=ldapuxprofile,ou=Profiles,ou=ldap-ux,o=hp
servicesearchdescriptor: passwd:ou=ldap-ux,o=hp?sub?(objectclass=posixaccount)
servicesearchdescriptor: shadow:ou=ldap-ux,o=hp?sub?(objectclass=shadowaccount)
...
```

Of course it is not desirable to use a privileged account as a proxy user. In order to use a non-privileged account as a proxy user the proxy user must be given Browse Rights to the OU the profile will be stored in. This can easily be done using ConsoleOne to assign Browse rights to the proxy user (cn=Proxy,ou=People,ou=ldap-ux,o=hp) for the OU (ou=Profiles,ou=ldap-ux,o=hp) where the profile will be created :

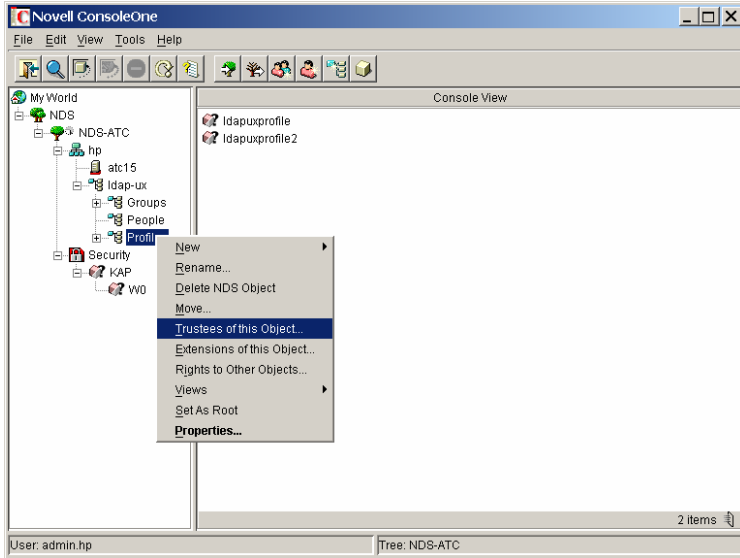


An ldapsearch on the ou=Profiles,ou=ldap-ux,o=hp will show 2 new ACL entries:

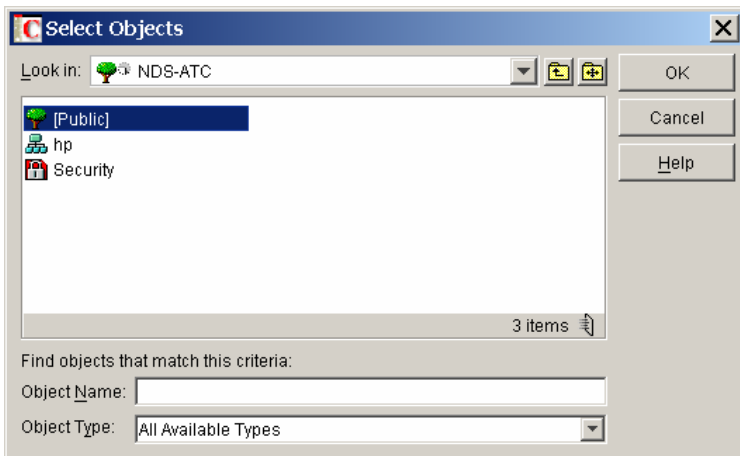
```
# ldapsearch -h ldap.hp.com -D "cn=proxy,ou=People,ou=ldap-ux,o=hp" -w PassWord \
-b "o=hp" cn=proxy ACL
dn: cn=Proxy,ou=People,ou=ldap-ux,o=hp
...
ACL: 1#subtree#ou=Profiles,ou=ldap-ux,o=hp#[Entry Rights]
ACL: 3#subtree#ou=Profiles,ou=ldap-ux,o=hp#[All Attributes Rights]
```

4.3.2 Deploying LDAP-UX without a Proxy User

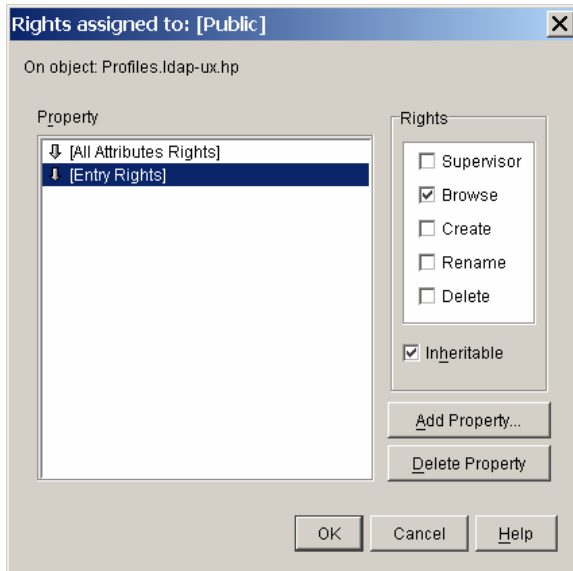
If a proxy user is not used access control on the OU the configuration profile will be stored must be opened up to allow anonymous (Public) Browse access. Again using ConsoleOne select the OU, right click, and select "Trustees of this Object"



When the properties window appears select “Add Trustee”, next select the “Public” object:



Now assign “Browse” Rights:



Now the configuration profile will be readable without having to authenticate to the Directory:

```
# ldapsearch -h ldap.hp.com -b "ou=profiles,ou=ldap-ux,o=hp" cn=ldapuxprofile
dn: cn=ldapuxprofile,ou=Profiles,ou=ldap-ux,o=hp
servicessearchdescriptor: passwd:ou=ldap-ux,o=hp?sub?(objectclass=posixaccount)
servicessearchdescriptor: shadow:ou=ldap-ux,o=hp?sub?(objectclass=shadowaccount)
servicessearchdescriptor: group:ou=ldap-ux,o=hp?sub?(objectclass=posixgroup)
...
```

New ACL's will be set on the profiles OU (ou=profiles,ou=ldap-ux,o=hp) object:

```
# ldapsearch -h ldap.hp.com -b "ou=ldap-ux,o=hp" ou=profiles ACL
dn: ou=Profiles,ou=ldap-ux,o=hp
ACL: 2#entry#ou=Profiles,ou=ldap-ux,o=hp#loginScript
ACL: 2#entry#ou=Profiles,ou=ldap-ux,o=hp#printJobConfiguration
...
ACL: 1#subtree#ou=Profiles,ou=ldap-ux,o=hp#[Entry Rights]
ACL: 1#subtree#[Public]#[Entry Rights]
ACL: 3#subtree#[Public]#[All Attributes Rights]
```

4.4 POSIX Attributes (RFC 2307)

Default installations of eDirectory do not have the schema extensions that support RFC 2307. The 8.6.2 distribution of eDirectory on Solaris includes both LDIF (LDAP Interchange Format) and SCH (Novell LDAP ASN1 Schema Definition) files that define the RFC 2307 object classes and attributes.

The schema files are found in the `/usr/share/nds-schema` directory on 8.6, `/usr/lib/nds-schema` on the 8.7 Linux distribution. The files related to RFC 2307 are:

rfc2307-usergroup.ldif	LDIF file for posixAccount and posixGroup Object classes and supporting attributes
rfc2307-usergroup.sch	SCH file for posixAccount and posixGroup Object classes and supporting attributes
rfc2307-nis.ldif	LDIF file for other NIS Map Object classes and supporting attributed
rfc2307-nis.sch	SCH file for other NIS Map Object classes and supporting attributed

There are several ways to extend the eDirectory Schema depending on the state of the Directory.

If the eDirectory has not yet been configured on your Solaris system using the `ndsconfig` program add the lines “`rfc2307-usergroup.sch`” & “`rfc2307-nis.sch`” (no quotes) to the `/usr/share/nds-schema/schema.cfg` file:

```
$ cat /usr/share/nds-schema/schema.cfg
# This is the current list of schema extensions.
# The ordering should not be changed because
# some schemas depend on the previous schema
# being already installed. If you have a new
# schema it should be added to the end of the
# list.
```

[Schema Extensions - Post DS]

```
nds500.sch
ldap.sch
ldapupdt.sch
nov_inet.sch
nwadmin.sch
ndscomm.sch
sas.sch
ndspki.sch
ndspkis.sch
masv.sch
rfc2307-usergroup.sch
rfc2307-nis.sch
```

When configuration/installation of eDirectory is done using the ndsconfig program the eDirectory Schema will automatically be extended to support the RFC 2307 object classes and attributes defined in the rfc2307-usergroup.sch and rfc2307-nis.sch files.

If the eDirectory has already been configured the same files can be used to extend the schema of a running eDirectory Server. To extend the schema of a running eDirectory Server use the /usr/bin/ndssch program:

```
# ndssch cn=admin,o=hp /usr/share/nds-schema/rfc2307-usergroup.sch
Password:
Logging into the tree as "cn=admin.o=hp". Please Wait ...
```

Extending schema, For more details view schema extension logfile: /var/nds/schema.log

NDS schema extension complete.

```
# ndssch cn=admin,o=hp /usr/share/nds-schema/rfc2307-nis.sch
Password:
Logging into the tree as "cn=admin.o=hp". Please Wait ...
```

Extending schema, For more details view schema extension logfile: /var/nds/schema.log

NDS schema extension complete.

```
# ldapsearch -h ldap.hp.com -D "cn=admin,o=hp" -w PaSSworD -s base \
-b "cn=schema" objectclass=* |grep posix
objectClasses: ( 1.3.6.1.1.1.2.0 NAME 'posixAccount' DESC 'Standard ObjectClas
objectClasses: ( 1.3.6.1.1.1.2.2 NAME 'posixGroup' DESC 'Standard ObjectClass'
```

If your eDirectory is running on a Windows platform please refer to the Novell website, or see Appendix A.

4.5 Access Control

Access control for the LDAP-UX Configuration Profile has been discussed in the proxy user section above.

The LDAP-UX product must be able to retrieve the POSIX Attributes of a user and group in order to allow eDirectory based users access to the HP-UX system. Whether or not a proxy user is configured it's likely that ACL modifications will need to be made. To determine if ACL changes need to be made use the `ldapsearch` command to "simulate" a lookup that LDAP-UX will make. The syntax of the command will differ based on your client binding (Proxy/Anonymous) configuration.

For **Anonymous** configurations:

Search for Users:

```
ldapsearch -h <Directory Server> -b <Search Base> \
    (&(uid=<UID>)(objectclass=posixaccount)) <attribute list>
```

Search for Groups:

```
ldapsearch -h <Directory Server> -b <Search Base> \
    (&(cn=<Group Name>)(objectclass=posixGroup)) <attribute list>
```

Example:

User Search:

```
# ldapsearch -h ldap.hp.com -b "o=hp" (&(uid=ldapux)(objectclass=posixaccount)) \
    homeDirectory gidNumber uidNumber loginshell uid
dn: cn=ldapux,ou=People,ou=ldap-ux,o=hp
loginshell: /usr/bin/ksh
homeDirectory: /home/ldapux
gidNumber: 40
uidNumber: 1001
uid: ldapux
```

Group Search:

```
# ldapsearch -h ldap.hp.com -b "o=hp" (&(cn=ldapusrs)(objectclass=posixGroup)) \
    gidnumber uniqueMember memberUid cn
dn: cn=ldapusrs,ou=Groups,ou=ldap-ux,o=hp
gidnumber: 40
uniqueMember: cn=ldapux,ou=People,ou=ldap-ux,o=hp
cn: ldapusrs
```

For **proxy user** configurations:

```
ldapsearch -h <Directory Server> -D <Proxy User DN> -w <Proxy User Password> \
    -b <Search Base> uid=<UID> <attribute list>
```

Example:**User Search:**

```
# ldapsearch -h ldap.hp.com -D "cn=proxy,ou=People,ou=ldap-ux,o=hp" -w PassWorD \
-b "o=hp" (&(uid=ldapux)(objectclass=posixaccount)) \
  homeDirectory gidNumber uidNumber loginShell uid
dn: cn=ldapux,ou=People,ou=ldap-ux,o=hp
loginShell: /usr/bin/ksh
homeDirectory: /home/ldapux
gidNumber: 40
uidNumber: 1001
uid: ldapux
```

Group Search:

```
# ldapsearch -h ldap.hp.com -D "cn=proxy,ou=People,ou=ldap-ux,o=hp" -w PassWorD \
-b "o=hp" (&(cn=ldapusrs)(objectclass=posixGroup)) \
  gidnumber uniqueMember memberUid cn
dn: cn=ldapusrs,ou=Groups,ou=ldap-ux,o=hp
gidnumber: 40
uniqueMember: cn=ldapux,ou=People,ou=ldap-ux,o=hp
cn: ldapusrs
```

If the attributes are displayed by the appropriate `ldapsearch` command then LDAP-UX has at least Read/Browse access to the POSIX Attributes.

For a user entry the following attributes must be viewable:

loginShell
gidNumber
uidNumber
uid

The following attributes “should” be viewable:

gecos
homeDirectory

For a group entry the following attributes must be viewable:

cn
gidNumber

The following attributes “should” be viewable (although if they are not viewable HP-UX will consider the group to have no members, which is valid but not very useful):

uniqueMember
memberUid

If the attributes are not readable (and are set on the user) then access control must be set such that either the proxy user has read/browse access to them or anonymous (Public) users have read/browse access to them. Access control can be set using `ConsoleOne`, or using an `ldapmodify` command on the appropriate object.

While it is possible to set ACL's on each user and group entry it is not recommend. Setting ACL's on every User and Group object can be a burden to administrator's, but more importantly adds additional overhead on the Directory Server which has to store and analyze

the additional information for each object. It is recommended that ACL's be set on the container(s) where users and groups are located.

Setting ACL's on the Container Object:

There are 2 options for allowing read/browse access on container objects:

1. Allow read/browse access on all objects and attributes in the container
2. Allow read/browse access on only the POSIX attributes in the container

NOTE: The procedures for setting ACL's are the same for both proxy and anonymous access. For anonymous access use [Public] for the account that's being given access, for proxy user configuration use the DN (Distinguished Name) of the proxy user configured for LDAP-UX.

eDirectory ACL Primer:

ACL's on objects in eDirectory contain the following fields (separated by a '#'):

- **Privileges** – A bitwise OR'd value of the following:

Value	Individual Attributes	Entry Rights
1	Compare	Browse
2	Read	Add
4	Write, Add & Delete	Delete
6	Add/Delete Self	Rename
16	N/A	Supervisory
32	Supervisory	N/A

- **Scope:**
 - Entry – The ACL is applied to the entry the ACL is defined on
 - Subtree – The ACL is applied to the entry the ACL is defined on as well as entries defined below this container
- The DN (**Distinguished Name**) of the object in which the ACL entry applied to. In addition to the Users DN the following pre-defined values can be used:
 - [Public] – All objects within the Directory Tree
 - [Self] – The User authenticated in the current connection
 - [Creator] - The user who created the object
 - [Inheritance Mask] - Filters or masks the privileges granted to an object.
 - [Root] - Denotes the directory tree root object
- The **attribute** the ACL is being applied to. Other pre-defined values are:
 - [Entry Rights] - Privileges apply to the entire object, rather than an attribute.
 - [All Attributes Rights] - Privileges apply to all attributes of the object

The following ACL grants Compare and Read access on the myInfo attribute to the user Joe Cool (uid=jcool,ou=People,o=Acme) for all object in the container (OU) this ACL is set on:

ACL: 3#subtree#uid=jcool,ou=People,o=Acme#myInfo

For more details on ACL's in eDirectory see:

<http://developer.novell.com/research/sections/netmanage/dirprimer/2001/june/p010601.htm>

Option 1: Setting ACL's to allow read/browse on ALL attributes in a container

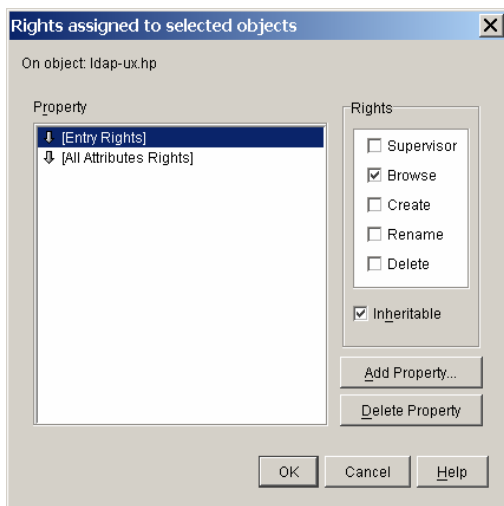
The following ACL's must be set on the outermost container where users and groups will be stored (remember to replace "[Public]" with the proxy users DN when using a Proxy configuration for LDAP-UX):

ACL: 1#subtree#[Public]#[Entry Rights]

ACL: 3#subtree#[Public]#[All Attributes Rights]

This can be done using ConsoleOne:

- Right click on the container object
- Select "Trustees of this Object"
- Click the Add Trustee button
- Choose [Public]
- Select "Browse" under Entry Rights
- Select "Compare" and "Read" under All Attributes Rights



Or use ldapmodify:

```
# ldapmodify -h ldap.hp.com -D "cn=admin,o=hp" -w PaSSwOrd
dn: ou=ldap-ux,o=hp
changetype: modify
add: ACL
ACL: 1#subtree#[Public]#[Entry Rights]
```

modifying entry ou=ldap-ux,o=hp

```
dn: ou=ldap-ux,o=hp
changetype: modify
add: ACL
ACL: 3#subtree#[Public]#[All Attributes Rights]
```

modifying entry ou=ldap-ux,o=hp

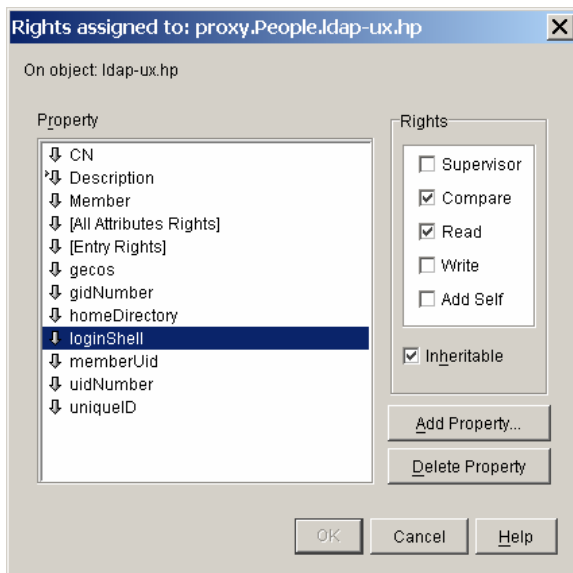
<Ctrl-D>

Option 2: Setting ACL's to allow read/browse of POSIX Account and Groups attributes in a container

The following ACL's must be set on the outermost container where users and groups will be stored (remember to replace the proxy users DN with "[Public]" when using a Anonymous binding for LDAP-UX):

```
ACL: 0#subtree#[Public]#[All Attributes Rights]
ACL: 1#subtree#cn=proxy,ou=People,ou=ldap-ux,o=hp#[Entry Rights]
ACL: 0#subtree#cn=proxy,ou=People,ou=ldap-ux,o=hp#[All Attributes Rights]
ACL: 3#subtree#cn=proxy,ou=People,ou=ldap-ux,o=hp#gidNumber
ACL: 3#subtree#cn=proxy,ou=People,ou=ldap-ux,o=hp#cn
ACL: 3#subtree#cn=proxy,ou=People,ou=ldap-ux,o=hp#gecos
ACL: 3#subtree#cn=proxy,ou=People,ou=ldap-ux,o=hp#loginShell
ACL: 3#subtree#cn=proxy,ou=People,ou=ldap-ux,o=hp#uidNumber
ACL: 3#subtree#cn=proxy,ou=People,ou=ldap-ux,o=hp#uid
ACL: 3#subtree#cn=proxy,ou=People,ou=ldap-ux,o=hp#homeDirectory
ACL: 3#subtree#cn=proxy,ou=People,ou=ldap-ux,o=hp#description
ACL: 3#subtree#cn=proxy,ou=People,ou=ldap-ux,o=hp#memberUid
ACL: 3#subtree#cn=proxy,ou=People,ou=ldap-ux,o=hp#uniqueMember
```

Again these ACL's can be set using ConsoleOne using the same procedure above:



or using `ldapmodify`. When making multiple modifications to a single object it is recommended that the `/opt/ldapux/bin/ldapentry` tool is used. The `ldapentry` tool retrieves the given object and launches a `vi` window with the object's attributes in LDIF format. Simply add the above ACL entries to the file and exit (with save: `"ZZ"` or `":wq!"`). You will then be asked to confirm your desire to modify the entry.

```
ldapentry -m -D <Admin DN> -w <Admin Password> <Container DN>
```

```
# /opt/ldapux/bin/ldapentry -m -D "cn=admin,o=hp" -w PasSWord ou=ldap-ux,o=hp
<At this point a vi (or the editor defined in your EDITOR environment variable) session is
started>
```

<After editing and saving the file answer yes when prompted>

Apply the changes and replace entry in directory? (y/n):y
modifying entry ou=ldap-ux,o=hp

Modified.

#

Granting Users Write/Update Privileges to their Objects

If it is desirable to allow users to update appropriate attributes on their object in the Directory ACL's must be set to allow this operation.

WARNINIG: Giving users the ability to modify some attributes can result in un-authorized privileged access. Use extreme care when granting write access to non-privileged users.

To allow users to update their loginShell attribute, an ACL on the users object must be set granting the user write access to the attribute. For example:

ACL: ~~4#entry#~~cn=ldapux, ou=People, ou=ldap-ux, o=hp#loginShell

In eDirectory 8.7 a new ACL subject name "[This]" has been added. Setting the following ACL on the container (OU) where users are located will grant write access on the loginShell attribute to the user. (**NOTE:** This has not been verified since all testing was done on 8.6.2)

ACL: ~~4#subtree#~~[This]#loginShell

4.6 Password Management

HP-UX systems running LDAP-UX with eDirectory can take advantage of all of the Password Policies enforced by eDirectory through LDAP. In order for an HP-UX user to change their password in eDirectory they must be given write access to the PasswordManagement attribute. This can be done using ldapmodify:

On releases prior to 8.7 add the following ACL to each user:

ACL: 4#subtree#<User DN>#passwordManagement

Example:

```
# ldapmodify -h ldap.hp.com -D "cn=admin,o=hp" -w PaSSwOrd
dn: cn=Joe,ou=People,ou=ldap-ux,o=hp
changetype: modify
add: ACL
ACL: 4#subtree#cn=Joe,ou=People,ou=ldap-ux,o=hp#passwordManagement
```

On 8.7 add the following ACL on the root container where users are stored:

ACL: 4#subtree#[This]#passwordManagement

Example:

```
# ldapmodify -h ldap.hp.com -D "cn=admin,o=hp" -w PaSSwOrd
dn: ou=People,ou=ldap-ux,o=hp
changetype: modify
add: ACL
ACL: 4#subtree#[This]#passwordManagement
```

Below is a list of Password Policy features that have been verified to work with LDAP-UX and eDirectory:

- Changing a user's password in eDirectory using the HP-UX passwd command.
- When a user's password has expired in eDirectory the user is not prompted to change his/her password at login time. The user will be unable to login to an HP-UX system until their password is changed in eDirectory.
- Users with expired eDirectory passwords are not able to login to an HP-UX system (Password expirations are enforced).
- Users with expired eDirectory passwords are allowed to login to an HP-UX system if they have 1 or more "Grace Logins" remaining (loginGraceRemaining attribute.)

4.7 Migrating and Managing Users and Groups

It's likely that some users and/or groups will already have entries in the Directory. For these users to be recognized by an HP-UX system with LDAP-UX enabled they will need to have POSIX attributes added to their entry. It's also possible that new UNIX users and/or Groups will need to be added into the eDirectory. These users will not only need to have the POSIX attributes defined, but they will need additional attributes in which eDirectory requires.

4.7.1 Migrating UNIX Users and Groups into eDirectory

The LDAP-UX product includes migration scripts (`/opt/ldapux/migrate`) that can be used to convert local `passwd` & `group` files or NIS `passwd` & `group` maps into an LDIF file that can be used to add users into the Directory. The 2 scripts available for migrating users and groups are `migrate_passwd.pl` & `migrate_group.pl` respectively. Both scripts take a file as input in `passwd/group` format and an optional output file as the second parameter, if no output file is specified the scripts will print the ldif file to `stdout`. There are a few assumptions made by these scripts:

- The environment variable `LDAP_BASEDN` is set to indicate where the Users and Groups will be created in
- All Users will be created in the container `"ou=People,$LDAP_BASEDN"` and groups will be created in `"ou=Group,$LDAP_BASEDN"`. The `"People"` & `"Group"` values can be changed by editing the following lines in the file `migrate_common.ph`:


```
$NAMINGCONTEXT{passwd}    = "ou=People";
$NAMINGCONTEXT{group}     = "ou=Group";
```

Migrating Users

The standard LDAP-UX migration scripts do not produce an LDIF entry for users that can be loaded into eDirectory. There are 2 modifications that need to be made to the `migrate_passwd.pl` script to produce a usable LDIF file:

1. Remove/comment out the following line:

```
print $HANDLE "objectClass: account\n";
```

The reason this needs to be removed is because eDirectory does not, by default, define the `"account"` objectClass from RFC 1274 in it's schema.

2. Add the following 2 lines just above or below the line commented out or removed in #1 above:

```
print $HANDLE "objectClass: inetOrgPerson\n";
print $HANDLE "sn: $sn\n";
```

These 2 lines must be added because eDirectory requires that the users object have the Objectclass of `inetOrgPerson` (or `person`) which is actually a mapping to an NDS User class. The SN (Surname) attribute is a mandatory attribute for the `inetOrgPerson`.

The following is a diff output between the standard migrate_passwd.pl script (v 1.4 from LDAP-UX 3.02) and a modified script:

```
# diff migrate_passwd.pl edir/migrate_passwd.pl
113c113,115
<   print $HANDLE "objectClass: account\n";
---
>   print $HANDLE "objectClass: inetOrgPerson\n";
>   print $HANDLE "sn: $sn\n";
>   #print $HANDLE "objectClass: account\n";
```

The following LDIF output was generated using the password entry:

```
ldaptest:CIBxqVP4kZfo:999:20:,,,:/home/ldaptest:/usr/bin/sh

# /opt/ldapux/migrate/edir/migrate_passwd.pl /tmp/passwd.tmp
dn: uid=ldaptest,ou=People,ou=ldap-ux,o=hp
uid: ldaptest
cn: ldaptest
objectClass: top
objectClass: inetOrgPerson
sn: ldaptest
objectClass: posixAccount
userPassword: {crypt}CIBxqVP4kZfo
loginShell: /usr/bin/sh
uidNumber: 999
gidNumber: 20
homeDirectory: /home/ldaptest
gecos: ,,,
```

Migrating Groups

The standard LDAP-UX group migration script does not produce an LDIF entry that can be loaded into eDirectory. There is 1 modification that needs to be made to the migrate_group.pl script to produce a usable LDIF file:

Add the following line in the **dump_group()** subroutine:

```
print $HANDLE "objectClass: groupOfNames\n";
```

The following is a diff output between the standard migrate_group.pl script (v 1.3 from LDAP-UX 3.01) and a modified script:

```
# diff migrate_group.pl edir/migrate_group.pl
68a69
>   print $HANDLE "objectClass: groupOfNames\n";
```

The following is the result of running the migrate_group.pl script against the group entry:

```
ldapusers::22:ldaptest,ldapux

# ./migrate_group.pl /tmp/group.tmp
dn: cn=ldapusers,ou=Group,ou=ldap-ux,o=hp
```

```
objectClass: groupOfNames
objectClass: posixGroup
objectClass: top
cn: ldapusers
gidNumber: 22
memberUid: ldaptest
memberUid: ldapux
```

4.7.2 Managing UNIX Users and Groups in eDirectory

Most likely users and groups will already exist in your Directory. In this case it is just a matter of adding POSIX attributes to that object. In general, regardless of the repository, user and group management is done using the tools Administrators are most comfortable with; using eDirectory to store UNIX users and Groups will be no differently.

There are many ways to add POSIX attributes to a user or group, which one is the right way is a matter of preference. Below are a few examples of how to add POSIX attributes to an existing Users object.

Adding POSIX Attributes to a User's object using ldapmodify

```
# /opt/ldapux/bin/ldapmodify -h ldap.hp.com -D cn=admin,o=hp -w pAsSWorD
dn: cn=Joe,ou=People,ou=ldap-ux,o=hp
changetype: modify
add: objectclass
objectclass: posixAccount
-
add: loginShell
loginShell: /bin/sh
-
add: homeDirectory
homeDirectory: /home/ldapux
-
add: gidNumber
gidNumber: 40
-
add: uidNumber
uidNumber: 10099
```

modifying entry cn=Joe,ou=People,ou=ldap-ux,o=hp

```
# ldapsearch -h ldap.hp.com -b "o=hp" uid=jcool
dn: cn=Joe,ou=People,ou=ldap-ux,o=hp
loginShell: /bin/sh
homeDirectory: /home/ldapux
gidNumber: 40
uidNumber: 10099
uid: jcool
sn: Cool
objectClass: inetOrgPerson
objectClass: organizationalPerson
```

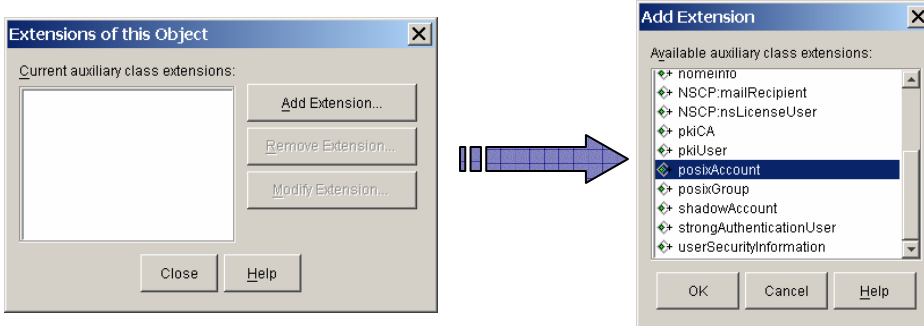
```
objectClass: person
objectClass: ndsLoginProperties
objectClass: top
objectClass: posixAccount
cn: Joe
```

```
# pwget -n jcool
jcool:*:10099:40::/home/ldapux:/bin/sh
```

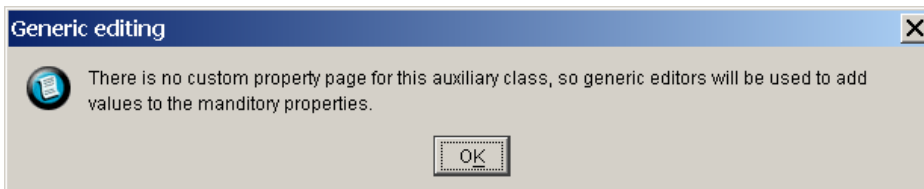
Adding POSIX Attributes to a User's object using ConsoleOne

Novell provides a GUI based tool called ConsoleOne that can be used to manage and configure the Directory Server as well as create, modify & delete User and Group objects (among other tasks).

By default ConsoleOne does not have a built-in panel that can be used to configure the POSIX attributes of a user or group. It is still possible to add the POSIX attributes by Right clicking on the users object and selecting "Extensions of this Object". A window will appear, select "Add Extension..", select posixAccount (or posixGroup for a group) and hit OK:

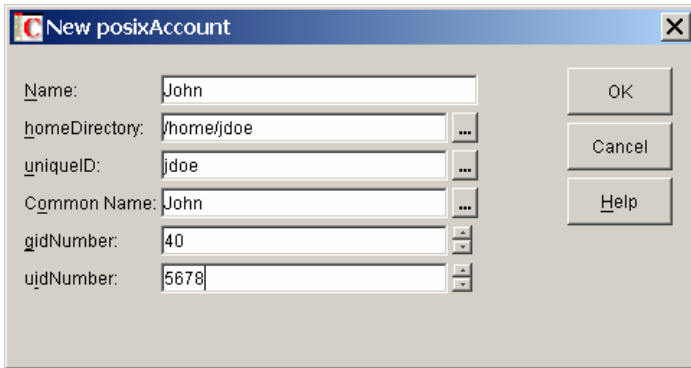


A pop-up window will appear stating there is no custom property page for this auxiliary class and that a generic editor will be used to add values to the **mandatory** properties.



It's important to note that only the mandatory fields will be presented. One field that is not listed as Mandatory in the RFC2307 that must be filled in manually is loginShell. If the attribute, loginShell, was listed as Mandatory and not Optional when eDirectory's schema was updated to support RFC 2307 this attribute would have been listed. (Hint: If you plan on using the generic editor in ConsoleOne modify the RFC 2307 SCH file to make loginShell a Mandatory attribute).

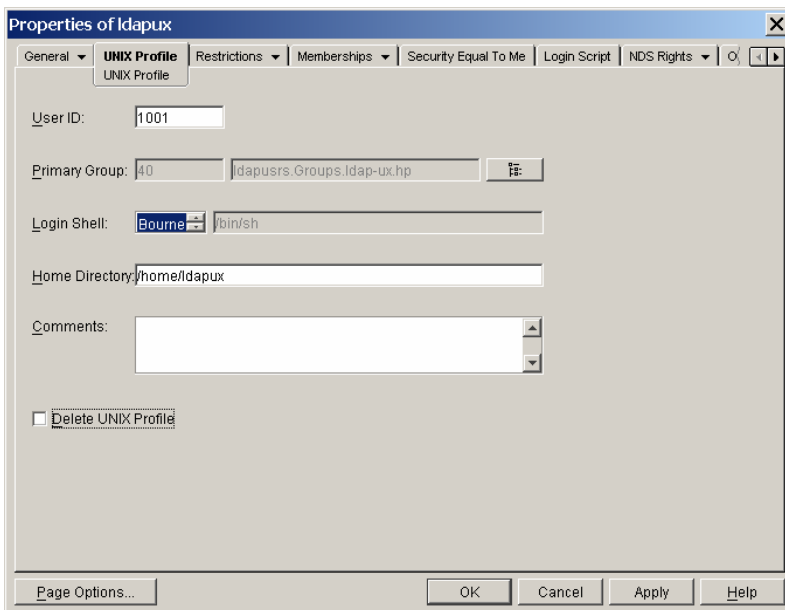
Fill in all of the fields and select OK:



Novell also provides a Snap-In for ConsoleOne to Manage UNIX Accounts. This Snap-In must be downloaded from Novell:

<http://support.novell.com/servlet/filedownload/pub/c1unx85a.exe>

Using this Snap-In to ConsoleOne UNIX Attributes can be managed directly from the UNIX Profile Pane:



Note that the users **Primary Group** must exist in eDirectory otherwise the ConsoleOne Properties will complain about not having the Primary Group set.

4.8 Helpful Links

- http://www.novell.com/coolsolutions/nds/features/tips/tru64_edir.html
- <http://developer.novell.com/research/appnotes/2002/june/02/a020602.htm>
- <http://developer.novell.com/research/sections/netmanage/dirprimer/2001/june/p010601.htm>

5 Oracle Internet Directory

Oracle Internet Directory (OID) is part of [Oracle9i Database Release 2](#) and is available on several platforms (HP-UX, Windows 2000, Linux, etc.). The following information was obtained while testing with Oracle Internet Directory version 9.2 on HP-UX 11.0. There have been successful deployments of LDAP-UX with Oracle Internet Directory on Windows as well. The following chapter discusses how to configure Oracle Internet Directory to store UNIX users and groups. While it is possible to store other NIS Maps in Oracle Internet Directory this document does not discuss it.

To successfully integrate LDAP-UX with OID perform the following steps (discussed in further detail later in this chapter):

1. Prepare OID for LDAP-UX
 - a. Modify supportedLdapVersion attribute
 - b. Create a proxy user (if required)
 - c. Extend the Directory schema to support RFC 2307
 - d. Modify ACL's (if required)
 - e. Modify the Directory to index gidnumber and uidnumber
 - f. Configure password policies (if required)
2. Run LDAP-UX setup tool
3. Configure `/etc/pam.conf` & `/etc/nsswitch.conf` (see `pam.ldap` and `nsswitch.ldap` for examples)
4. Test the LDAP-UX configuration (you'll need at least 1 POSIX user/group configured in OID)
5. Migrate and/or configure UNIX users and groups

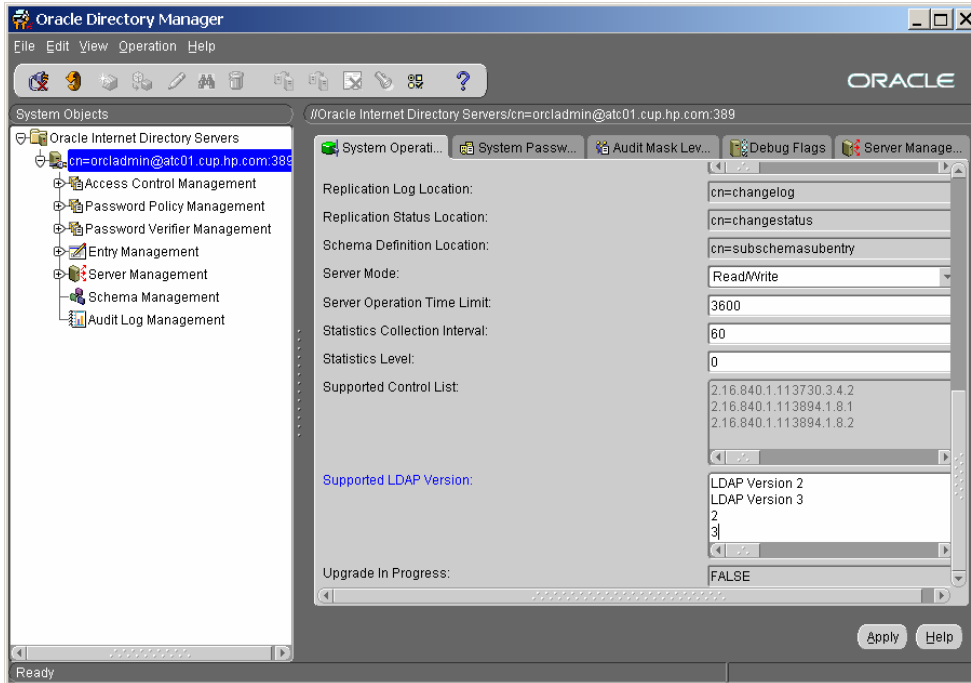
5.1 Preparing Oracle Internet Directory for LDAP-UX Integration

Prior to configuring LDAP-UX on your HP-UX system there are a few issues to be addressed within Oracle Internet Directory

5.1.1 Updating the LDAP Operational Attribute supportedLDAPVersion

In order to run the LDAP-UX setup tool successfully with Oracle Internet Directory the operational attribute supportedLDAPVersion must be updated. When the LDAP-UX setup tool is initially run it checks to see if the Directory Server being used supports LDAP Version 3. To determine what LDAP protocol versions the Directory Server supports the setup tool examines the LDAP operational attribute supportedLDAPVersion, defined in RFC 2252 (<http://www.ietf.org/rfc/rfc2252.txt>). According to RFC 2252 the supportedLDAPVersion attribute uses INTEGER syntax, where the value is encoded "as the decimal representation of their values". When the supportedLDAPVersion attribute is retrieved from the Oracle Internet Directory the string "LDAP Version 3" is returned. Because the LDAP-UX setup tool is expecting an integer value the comparison fails, and the setup tool concludes that the Directory Server does not support LDAP Version 3. This problem has been documented and reported to Oracle in Bug Number: 2677668. When this is addressed by Oracle the following step will no longer be necessary.

The value of the supportedLDAPVersion attribute is modifiable, and multi-valued. While it should be possible to change the value from “LDAP Version 3” to “3” the affect on other applications are unknown, therefore adding 2 more values for the supportedLDAPVersion attribute was done to allow for backwards compatibility. According to the Oracle Internet Directory Administration Guide it should be possible to modify this attribute using either ldapmodify or the Oracle Directory Manager configuration tool. Unfortunately neither of these tools worked consistently when adding the additional values. In some case’s, usually the 1st time a modification was attempted, the Oracle Directory Manager configuration tool was able to modify the supportedLDAPVersion attribute:



The more reliable procedure to modify the supportedLDAPVersion attribute is to modify the Oracle Database directly. The following script was used to add the values “2” and “3” to the supportedLDAPVersion attribute:

```
#!/usr/bin/sh
#:
```

```
echo ">> Modify supportedLDAPVersion attributes.."
```

```

${ORACLE_HOME}/bin/sqlplus system/<SYSTEM Password>@<ORACLE SID> <<!
insert into ods.ds_attrstore (entryid, attrname, attrval, attrkind, attrver) values (1,
'supportedLDAPversion', '2', 'u', NULL);
insert into ods.ds_attrstore (entryid, attrname, attrval, attrkind, attrver) values (1,
'supportedLDAPVersion', '3', 'u', NULL);
COMMIT;
exit
!
```

Output:

```
$/tmp/mod_ldap_version_sql.sh
>> Modify supportedLDAPVersion attributes..

SQL*Plus: Release 9.2.0.1.0 - Production on Fri Dec 6 11:53:16 2002

Copyright (c) 1982, 2002, Oracle Corporation. All rights reserved.
```

```
Connected to:
Oracle9i Enterprise Edition Release 9.2.0.1.0 - 64bit Production JServer Release 9.2.0.1.0 -
Production
```

```
SQL>
1 row created.
```

```
SQL>
1 row created.
```

```
SQL>
Commit complete.
```

```
SQL> Disconnected from Oracle9i Enterprise Edition Release 9.2.0.1.0 - 64bit Production
JServer Release 9.2.0.1.0 - Production
```

--

NOTE: It was necessary to stop and restart the OID Ldap Server (oidldapd) in order for the changes to show up when using ldapsearch.

Before running the LDAP-UX setup tool verify that the supportedLDAPversion attribute has been updated to include the value '3'. Use the following command:

```
ldapsearch -h <OID Host> -s base -b "" objectclass=* supportedLDAPversion
The output should look like:
```

```
# ldapsearch -h atc01.cup.hp.com -s base -b "" objectclass=* supportedLDAPversion
dn:
supportedldapversion: LDAP Version 2
supportedldapversion: LDAP Version 3
supportedldapversion: 2
supportedldapversion: 3
```

IMPORTANT:

Make sure that the above ldapsearch command returns the supportedldapversion attributes, otherwise the LDAP-UX setup tool will not work, reporting the error:

“The Directory Server does not support LDAP v3!”

In some cases executing the above ldapsearch command **without** specifying the supportedLDAPversion attribute (telling ldapsearch to return all attributes):

```
# ldapsearch -h <OID Host> -s base -b "" objectclass=*
would return the supportedLDAPversion attribute while specifying the
supportedLDAPversion attribute to be returned:
# ldapsearch -h <OID Host> -s base -b "" objectclass=* supportedLDAPversion
did not actually return the value. If this occurs contact your local Oracle Support contact.
```

5.1.2 Configure a Proxy User

In a default Oracle Internet Directory installation/configuration a proxy user is NOT required for LDAP-UX to function properly. It is possible to modify the Access Control in the Oracle Internet Directory such that a proxy user is required for LDAP-UX. For more details on configuring proxy user access see the section title “*Proxy Users*” below.

5.1.3 Extend the Oracle Internet Directory Schema for RFC2307 (NIS/POSIX Attributes)

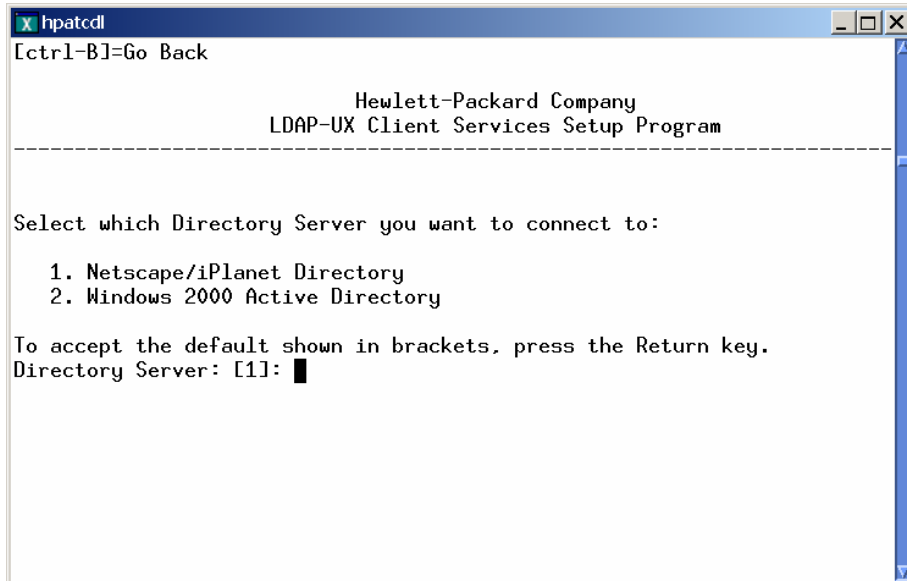
While it’s not necessary to extend the OID schema prior to configuring LDAP-UX it must be done prior to adding POSIX/UNIX users (or adding attributes to existing OID users) to the Directory. See the section below titled “*Posix Attributes (RFC2307)*” for more information.

5.1.4 Creating a Password Policy

If the Directory Server does not already have a password policy configured, or HP-UX users will be located outside the scope of existing password policies, it may be necessary to create a password policy before migrating users into the Directory. If a new password policy is required it must be created prior to migrating users into the Directory due to a defect (Oracle Bug 2509657) with Oracle Internet Directory. See Section 5.6 “Password Management” for details on creating a Password Policy.

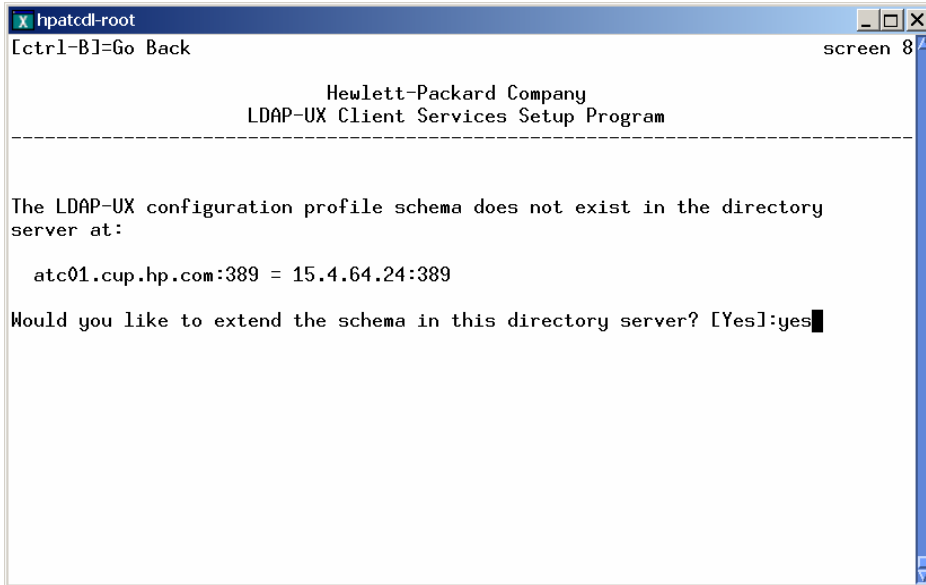
5.2 Profile Schema

When configuring LDAP-UX 3.0 with Oracle Internet Directory the setup tool can be used to successfully extend the schema to support the LDAP-UX configuration profile schema (DUAConfigProfile). When running the setup tool you will be asked to select the Directory Server to connect to:

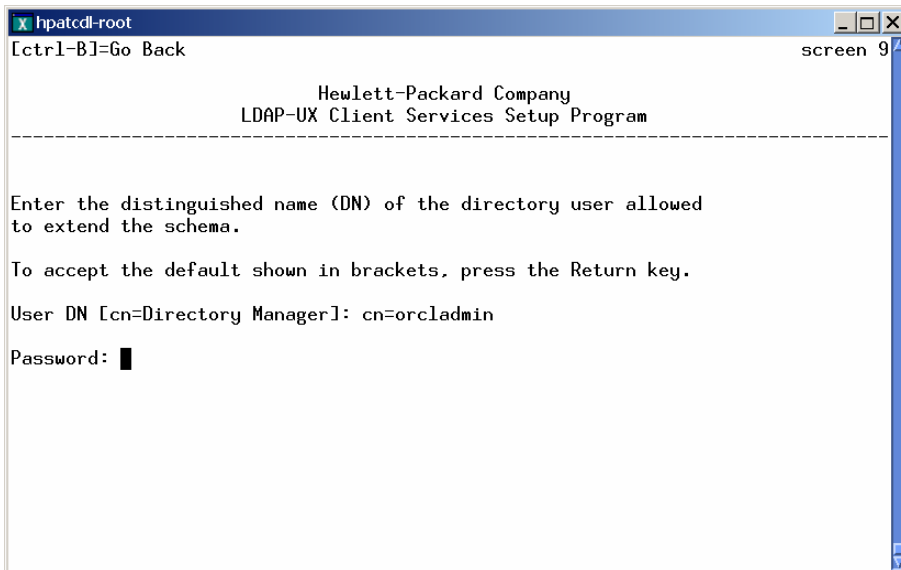


Select 1 (Netscape/iPlanet Directory).

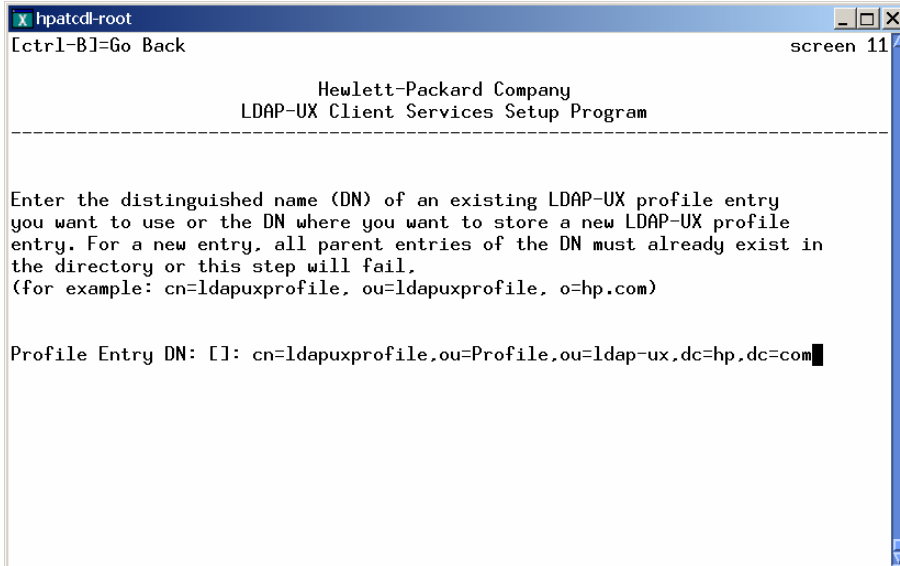
The first time LDAP-UX is configured with Oracle Internet Directory it will be required that the Directories schema be extended to support the LDAP-UX configuration profile. The setup tool will determine that the Directory Server has not been extended to support this new objectclass and will ask the user if the schema should be extended:



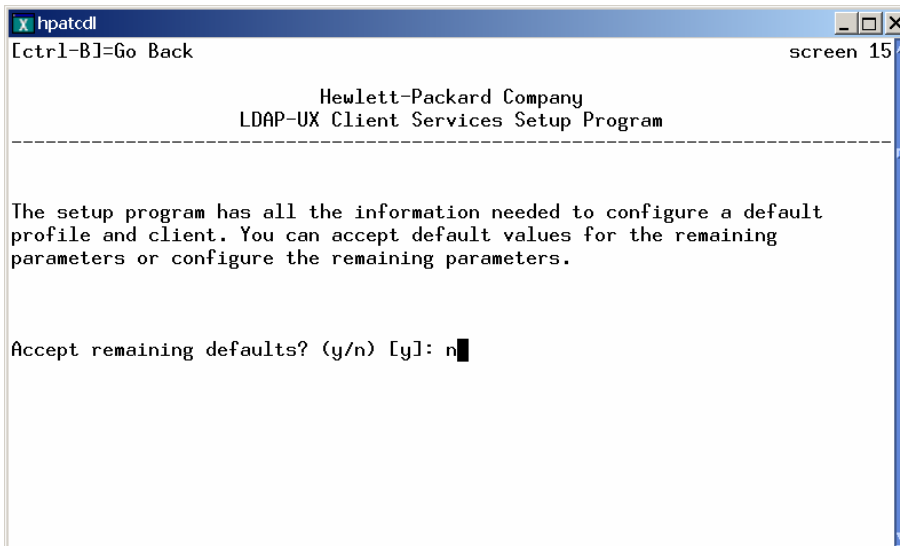
After entering the Directory Manager's Distinguished Name and Password:



setup will prompt the user for the location in the Directory where the configuration profile should be stored:

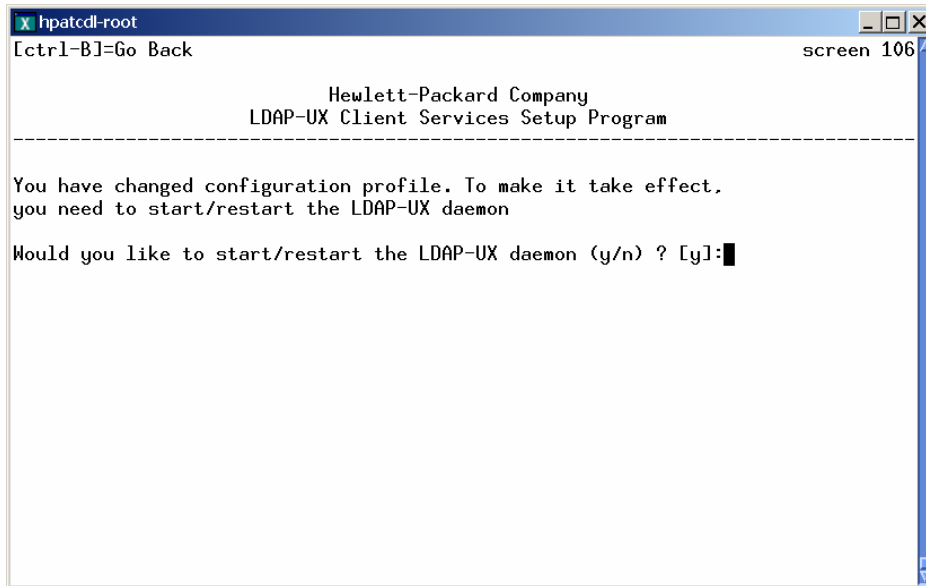


Follow the instructions as if you were configuring LDAP-UX for the Netscape/iPlanet Directory Server, **NOTE:** Digest-MD5 authentication has not been tested with Oracle Internet Directory. If you plan on configuring a proxy user answer NO when asked to “Accept remaining defaults?”:



otherwise answer y[es].

After the profile is created in the Directory setup will prompt the user to start/restart the ldapclntd:



```
hpatcdl-root
[ctrl-B]=Go Back                               screen 106

                Hewlett-Packard Company
                LDAP-UX Client Services Setup Program
-----
You have changed configuration profile. To make it take effect,
you need to start/restart the LDAP-UX daemon

Would you like to start/restart the LDAP-UX daemon (y/n) ? [y]:
```

After the `ldapclientd` is restarted the user is instructed to modify the `pam.conf` and `nsswitch.conf` files to LDAP:

```
Updated the LDAP-UX daemon configuration file
/etc/opt/ldapux/ldapclientd.conf
Restarted the LDAP-UX daemon!
```

To enable the LDAP Pluggable Authentication Module, save a copy of the file `/etc/pam.conf` then add `ldap` to it. See `/etc/pam.ldap` for an example.

To enable the LDAP Name Service Switch, save a copy of the file `/etc/nsswitch.conf` then add `ldap` to it. See `/etc/nsswitch.ldap` for an example.

LDAP-UX Client Services setup complete.

See the “Installing and Administering LDAP-UX Client Services” manual for more information on configuring PAM and NSSWITCH to use LDAP-UX.

5.3 Proxy Users

In the default configuration/installation of Oracle Internet Directory on HP-UX it is not necessary to configure a proxy user in order for LDAP-UX to function correctly. However, it is possible that access control to the Directory has been, or will be, modified such that anonymous access to the directory is disallowed.

NOTE: During a default installation of Oracle Internet Directory a “Proxy” user is created. The role of this user is different from the role of the proxy user configured for LDAP-UX. The role of the default proxy user (cn=proxy) defined by Oracle is:

“...typically employed in an environment with a middle tier such as a firewall. In such an environment, the end user authenticates to the middle tier. The middle tier then logs into the directory on the end user's behalf. A proxy user has the privilege to switch identities and, once it has logged into the directory, switches to the end user's identity. It then performs operations on the end user's behalf, using the authorization appropriate to that particular end user.”

5.3.1 Creating a Proxy User

A proxy user for LDAP-UX does not need to have POSIX attributes; in fact it's recommended that no POSIX attribute be added to the proxy user. To create a proxy user using the Directory Management tool right clicking on the location the proxy user is to be created in and select “Create”. Using the “Add” button select an inetOrgPerson. Fill in the required field cn and sn. Don't forget to fill in the userPassword field under the **Optional Properties** tab otherwise the user will be created without a password:

The screenshot shows a 'New Entry' dialog box with the following details:

- Distinguished Name:** cn=ldap-ux, ou=People, ou=ldap-ux, dc=hp, dc=com
- Object Classes:**

Name	Description
inetOrgPerson	
- Mandatory Properties:**
 - cn: ldap-ux
 - sn: Proxy
- Optional Properties:** (tab is visible but empty)

5.4 POSIX Attributes (RFC 2307)

The Oracle Internet Directory does not support the schema extensions defined in RFC2307, therefore the user must either extend the Directory's schema to support the standard RFC2307 objectclasses and attributes or find existing attributes that can be mapped to the Posix attributes. It's recommended that the Oracle Internet Directory's schema be extended to support RFC2307 to minimize configuration customization. A sample LDIF file containing the posixAccount and posixGroup objectclasses and supporting attributes has been provided in Appendix B.

Extending the Directory schema to support RFC2307 posixAccount and posixGroup objectclasses using ldapmodify:

```
# ldapmodify -D cn=orcladmin -w welcome -h atc01.cup.hp.com -f ./rfc2307_oid.ldif
modifying entry cn=subschemasubentry

modifying entry cn=subschemasubentry

modifying entry cn=subschemasubentry

...

modifying entry cn=subschemasubentry
```

Now verify:

```
# ldapsearch -D cn=orcladmin -w welcome -h atc01.cup.hp.com -s base \
-b cn=subschemasubentry cn=subschemasubentry | grep 1.3.6.1.1.1.1

attributetypes: ( 1.3.6.1.1.1.1.1      NAME 'gidNumber'          DESC 'An integer uniqu
attributetypes: ( 1.3.6.1.1.1.1.3      NAME 'homeDirectory'     DESC 'The abs
attributetypes: ( 1.3.6.1.1.1.1.5      NAME 'shadowLastChange'  EQUALITY tege
attributetypes: ( 1.3.6.1.1.1.1.7      NAME 'shadowMax'         EQUALITY integerMatch
attributetypes: ( 1.3.6.1.1.1.1.9      NAME 'shadowInactive'    EQUALITY integer
attributetypes: ( 1.3.6.1.1.1.1.11     NAME 'shadowFlag'        EQUALITY integerMat
attributetypes: ( 1.3.6.1.1.1.1.0      NAME 'uidNumber'         DESC 'An integer uniqu
attributetypes: ( 1.3.6.1.1.1.1.2      NAME 'gecos'             DESC 'The GECOS field;
attributetypes: ( 1.3.6.1.1.1.1.4      NAME 'loginShell'        DESC 'The path to the
attributetypes: ( 1.3.6.1.1.1.1.6      NAME 'shadowMin'         EQUALITY integerMa
attributetypes: ( 1.3.6.1.1.1.1.8      NAME 'shadowWarning'     EQUALITY integerMa
attributetypes: ( 1.3.6.1.1.1.1.10     NAME 'shadowExpire'      EQUALITY integerM
attributetypes: ( 1.3.6.1.1.1.1.12     NAME 'memberUid'        DESC 'The uids of the
```

Attributes that are added to the Oracle Internet Directory are not configured to be indexed by default. Before creating objects that contain new attributes that need to be indexed the Directory must be configured to index those attributes. In order for some HP-UX applications to function correctly with LDAP-UX two attributes, gidnumber and uidnumber, must be indexed. After the schema has been extended to support RFC2307 it must be configured to index gidnumber and uidnumber. Indexing existing attributes that have not been used by any object in the Directory can be done using ldapmodify. Once the attribute has been created it can NOT be indexed using the Oracle Internet Directory Management tool.

5.4.1 Indexing the gidnumber and uidnumber attributes using ldapmodify

Use the following LDIF file with ldapmodify to configure Oracle Internet Directory to index the gidnumber and uidnumber attributes:

```
#
# LDIF file used to index gidnumber and uidnumber attributes
#
dn: cn=catalogs
changetype: modify
add: orclindexedattribute
orclindexedattribute: uidnumber

dn: cn=catalogs
changetype: modify
add: orclindexedattribute
orclindexedattribute: gidnumber
```

Example ldapmodify command:

```
# ldapmodify -D cn=orcladmin -w welcome -h atc01.cup.hp.com -f ./index.ldif
modifying entry cn=catalogs

modifying entry cn=catalogs
```

NOTE: This ldapmodify command will take several seconds to complete.

Verify that both attributes are configured to be indexed:

```
# ldapsearch -D cn=orcladmin -w welcome -h atc01.cup.hp.com -s base -b cn=catalogs
cn=catalogs | grep -i idnumber
orclindexedattribute: gidnumber
orclindexedattribute: uidnumber
```

5.4.2 Indexing the gidnumber and uidnumber attributes after they have been used

If objects have been created using these attributes before the Directory was configured to index them it's necessary to use the catalog.sh script that is installed on the Oracle Internet Directory Server. The syntax for the command is:

```
catalog.sh -connect <Connect String> -add -attr <attribute>
```

After the catalog.sh script is run the Directory Server must be restarted in order for the new indexing to take effect.

Example:

```
# $ORACLE_HOME/ldap/bin/catalog.sh -connect ldapdb -add -attr gidnumber
```

This tool can only be executed if you know database user password for OiD

Enter OiD password ::

ADDING CATALOG INDEXES

This tool can only be executed if you know database user password for OiD

Enter OiD Password ::

Creating Catalog for Attribute :: gidnumber

Done

5.5 Access Control

Because information is stored in the Directory that controls access to resources, information and data on your LDAP-UX client it is extremely important to understand how access control is handled within the Directory Server. The Oracle Internet Directory, like most Directory Servers, allows ACI's to be placed on objects within the Directory. OID stores access control information in operational attributes (internal "attributes" that are not associated with an objectclass). The Oracle Internet Directory supports 2 types of operational attributes that are used to hold ACI's.

orclEntryLevelACI: Applies to only the entry that the ACI is associated with. Subentries below an entry that have an **orclEntryLevelACI** entry do not inherit these ACLs.

orclACI: Applies to the entry the ACL is associated with and any subentry of that object. While it is possible to use the **orclACI** attribute to specify an ACI for a specify entry it is recommended, for performance reasons, that **orclEntryLevelACI** be used.

Like most directory servers OID does enforce a default ACI if no ACI's have been defined for that entry and there is no ACI inherited from a higher level object. The default policy, defined in the Root/Default ACP, is:

```
orclaci: access to entry by * (browse,noadd,nodelete)
orclaci: access to attr=(userpkcs12, orclpkcs12hint, userpassword)
    by group="cn=oracleusersecurityadmins,cn=groups,cn=oraclecontext"
        (read,search,write,compare)
    by self (read,search,write,compare)
    by * (none)
orclaci: access to
    attr!=(userpkcs12, orclpkcs12hint, userpassword, orclpassword, orclpasswordverifier,
        orclstatsflag, orclstatsperiodicity)
    by * (read,search,compare)
orclaci: access to attr=(orclpassword, orclpasswordverifier)
    by self (read,search,write,compare)
    by * (none)
orclaci: access to attr=(orclstatsflag, orclstatsperiodicity)
    by dn="cn=emd admin,cn=oracle internet directory" (read,search,write,compare)
    by * (read,search)
```

It's important to note that this is the DEFAULT ACI set when OID is installed. If you are not working with a clean installation it's possible that additional ACI's have been set on the Directory. Verify the ACI's set on each leaf of the Directory tree before assuming the object(s) in question meet your security needs. A simple ldapsearch can be used to examine the ACI's of an object. Since it is possible to place ACI's on a container object as well as a leaf object you will need to start at the object in question and work backwards to the root of the directory tree.

Example 1:

Entry DN: cn=ldapuxprofile,ou=Profiles,ou=ldap-ux,dc=hp,dc=com

```
# ldapsearch -h ldap1 -D cn=orcladmin -w pass -b "dc=hp,dc=com" cn=ldapuxprofile orclaci
dn: cn=ldapuxprofile, ou=Profiles, ou=ldap-ux, dc=hp, dc=com
# ldapsearch -h ldap1 -D cn=orcladmin -w pass -b "dc=hp,dc=com" ou=Profiles orclaci
dn: ou=Profiles,ou=ldap-ux,dc=hp,dc=com
# ldapsearch -h ldap1 -D cn=orcladmin -w pass -b "dc=hp,dc=com" ou=ldap-ux orclaci
dn: ou=ldap-ux,dc=hp,dc=com
# ldapsearch -h ldap1 -D cn=orcladmin -w pass -b "dc=hp,dc=com" dc=hp orclaci
dn: dc=hp,dc=com
# ldapsearch ldap1 -D cn=orcladmin -w pass -b "" dc=com orclaci
dn: dc=com
```

In this example there are no orclaci attributes on the entire tree, therefore the default ACI would apply to the entry: cn=ldapuxprofile, ou=Profiles, ou=ldap-ux, dc=hp, dc=com

Example 2:

Entry DN: cn=secure_profile,ou=Secure Profiles,ou=ldap-ux,dc=hp,dc=com

```
# ldapsearch -h ldap1 -D cn=orcladmin -w pass -b "dc=hp,dc=com" "cn=secure_profile" orclaci
dn: cn=secure_profile, ou=Secure Profiles, ou=ldap-ux, dc=hp, dc=com
# ldapsearch -h ldap1 -D cn=orcladmin -w pass -b "dc=hp,dc=com" ou=Secure Profiles orclaci
dn: ou=Secure Profiles,ou=ldap-ux,dc=hp,dc=com
orclaci: access to attr=(*)
    by group= "cn=LDAP-UX Administrators,cn=Groups, dc=hp,dc=com"
        (read,search,compare,write)
    by group= "cn=LDAP-UX Proxy Users,cn=Groups, dc=hp,dc=com"
        (read,search,compare)
    by * (none)
orclaci: access to entry
    by group= "cn=LDAP-UX Administrators,cn=Groups, dc=hp,dc=com"
        (browse, add, delete)
    by group= "cn=LDAP-UX Proxy Users,cn=Groups, dc=hp,dc=com"
        (browse)
    by * (none)
```

In this example there is no ACI entry on the object (cn=secure_profile, ou=Secure Profiles, ou=ldap-ux, dc=hp, dc=com) but there is an ACI on the subtree the object resides in that allows members of the LDAP-UX Administrators group read/write/delete access to the subtree as well as giving members of the LDAP-UX Proxy Users group read/browse access to the entries in the subtree. All other users, including anonymous binds, are give no access to the subtree.

NOTE: In order for a group to be used in an ACI entry it must be marked as a “privileged group”. To set a group to be privileged the orclPrivilegeGroup object class must be added.

5.5.1 Configuration Profile

The ACIs that will be set on the configuration profile will depend on how the LDAP-UX client is configured to access the Directory. If no proxy user has been configured then the LDAP-UX client will use an anonymous bind when retrieving the configuration profile from the Directory.

Anonymous User

When using anonymous access with LDAP-UX the anonymous user must have read access to the profile entry. Using `ldapmodify` we add an ACI entry to the container where the profile configuration entries will be stored. This ACI will apply to all profile entries stored in this sub-tree unless an ACI is placed on the configuration profile object itself, or another container within this sub-tree:

```
# ldapmodify -h ldap1 -D cn=orcladmin -w ldapldap
dn: ou=Profiles,ou=ldap-ux,dc=hp,dc=com
changetype: modify
replace: orclaci
orclaci: access to entry by group= "cn=LDAP-UX Administrators,cn=Groups, dc=hp,dc=com"
(browse, add, delete) by * (browse)
orclaci: access to attr=(*) by group= "cn=LDAP-UX Administrators,cn=Groups,
dc=hp,dc=com" (read,search,compare,write) by * (read,search,compare)

modifying entry ou=Profiles,ou=ldap-ux,dc=hp,dc=com
```

Verify the ACI has been set properly using `ldapsearch`:

```
# ldapsearch -h ldap1 -D cn=orcladmin -w pass -b "dc=hp,dc=com" "ou=Profiles" orclaci
dn: ou=Profiles,ou=ldap-ux,dc=hp,dc=com
orclaci: access to entry by group= "cn=LDAP-UX Administrators,cn=Groups, dc=hp, dc=com"
(browse, add, delete) by * (browse)
orclaci: access to attr=(*) by group= "cn=LDAP-UX Administrators,cn=Groups, dc=hp,
dc=com" (read,search,compare,write) by * (read,search,compare)
```

Verify that anonymous users can view a profile stored in the Directory:

```
# ldapsearch -h ldap1 -b "ou=Profiles,ou=ldap-ux,dc=hp,dc=com" "cn=ldapuxprofile"
dn: cn=ldapuxprofile, ou=Profiles, ou=ldap-ux, dc=hp, dc=com
objectclass: top
objectclass: duaconfigprofile
cn: ldapuxprofile
preferredserverlist: 192.1.1.1:389
defaultsearchbase: ou=ldap-ux,dc=hp,dc=com
bindtimelimit: 5
...
```

Now verify that anonymous users AND a user that is NOT a member of the LDAP-UX Administrators group can NOT modify a profile entry:

1st check for any ACI's on the entry itself:

```
# ldapsearch -h ldap1 -b "ou=Profiles,ou=ldap-ux,dc=hp,dc=com" "cn=ldapuxprofile"
orclentrylevelaci orclaci
dn: cn=ldapuxprofile, ou=Profiles, ou=ldap-ux, dc=hp, dc=com
```

Now check for anonymous modification:

```
# ldapmodify -h ldap1
dn: cn=ldapuxprofile, ou=Profiles, ou=ldap-ux, dc=hp, dc=com
changetype: modify
replace: bindtimelimit
bindtimelimit: 555

modifying entry cn=ldapuxprofile, ou=Profiles, ou=ldap-ux, dc=hp, dc=com
ldap_modify: Insufficient access
```

Now check an authenticated, unauthorized, user modification:

```
# ldapmodify -D uid=ldapusr1,ou=People,ou=ldap-ux,dc=hp,dc=com -w ldap1
dn: cn=ldapuxprofile, ou=Profiles, ou=ldap-ux, dc=hp, dc=com
changetype: modify
replace: bindtimelimit
bindtimelimit: 555

modifying entry cn=ldapuxprofile, ou=Profiles, ou=ldap-ux, dc=hp, dc=com
ldap_modify: Insufficient access
```

Now for good measures test for authenticated, authorized user modification:

```
# ldapsearch -b "dc=hp,dc=com" "cn=LDAP-UX Administrators" uniquemember
dn: cn=LDAP-UX Adminstrators,cn=Groups, dc=hp,dc=com
uniquemember: cn=Joe,cn=Users,dc=hp,dc=com
uniquemember: cn=orcladmin, cn=Users, dc=hp,dc=com
```

Use “cn=Joe,cn=Users,dc=hp,dc=com” as the authorized user:

```
# ldapmodify -h ldap1 -D cn=Joe,cn=Users,dc=hp,dc=com -w ldap1234
dn: cn=ldapuxprofile, ou=Profiles, ou=ldap-ux, dc=hp, dc=com
changetype: modify
replace: bindtimelimit
bindtimelimit: 555

modifying entry cn=ldapuxprofile, ou=Profiles, ou=ldap-ux, dc=hp, dc=com
```

Verify change:

```
# ldapsearch -b "dc=hp,dc=com" "cn=ldapuxprofile" bindtimelimit
dn: cn=ldapuxprofile, ou=Profiles, ou=ldap-ux, dc=hp, dc=com
bindtimelimit: 555
```

If a proxy user is configured within LDAP-UX then it is possible to tighten the access control on the configuration profile (or the subtree configuration profiles will be stored) so that only proxy users can read the profile entry. Because the LDAP-UX client only needs to download the profile the proxy user should only be given read access to the profile entry. The ldapmodify program can be used to give the “LDAP-UX Proxy Users” group read access, along with giving the “LDAP-UX Administrators” group read/write access and no access to the anonymous users:

```
# ldapmodify -D cn=orcladmin -w ldapldap
dn: ou=Secure Profiles,ou=ldap-ux,dc=hp,dc=com
changetype: modify
replace: orclaci
orclaci: access to entry by group= "cn=LDAP-UX Adminstrators,cn=Groups, dc=hp,dc=com"
(browse, add, delete) by group= "cn=LDAP-UX Proxy Users,cn=Groups, dc=hp,dc=com"
(browse) by * (none)
orclaci: access to attr=(*) by group= "cn=LDAP-UX Adminstrators,cn=Groups,
dc=hp,dc=com" (read,search,compare,write) by group= "cn=LDAP-UX Proxy
Users,cn=Groups, dc=hp,dc=com" (read,search,compare) by * (none)

modifying entry ou=Secure Profiles,ou=ldap-ux,dc=hp,dc=com
```

Verify the ACI has been set:

```
# ldapsearch -D cn=orcladmin -w ldapldap -b "dc=hp,dc=com" "ou=Secure Profiles"
dn: ou=Secure Profiles,ou=ldap-ux,dc=hp,dc=com
orclaci: access to entry by group= "cn=LDAP-UX Adminstrators,cn=Groups, dc=hp, dc=com"
(browse, add, delete) by group= "cn=LDAP-UX Proxy Users,cn=Groups, dc=hp,dc=com"
(browse) by * (none)
orclaci: access to attr=(*) by group= "cn=LDAP-UX Adminstrators,cn=Groups,
dc=hp,dc=com" (read,search,compare,write) by group= "cn=LDAP-UX Proxy Users,
cn=Groups, dc=hp,dc=com" (read,search,compare) by * (none)
```

Check for ACI entries on the configuration profile object itself:

```
# ldapsearch -D cn=orcladmin -w ldapldap -b "ou=Secure Profiles, ou=ldap-ux,dc=hp,dc=com"
cn=secure_profile orclentrylevelaci orclaci
dn: cn=secure_profile, ou=Secure Profiles, ou=ldap-ux, dc=hp, dc=com
```

Check for anonymous read:

```
# ldapsearch -b "dc=hp,dc=com" cn=secure_profile
ldap_search: Insufficient access
```

Check for authenticated, unauthorized read access:

```
# ldapsearch -D uid=ldapusr1,ou=People,ou=ldap-ux,dc=hp,dc=com -w ldap1 -b
"dc=hp,dc=com" cn=secure_profile
ldap_search: Insufficient access
```

Check for authenticated, authorized read access:

```
# ldapsearch -b "dc=hp,dc=com" "cn=LDAP-UX Proxy Users" uniquemember
dn: cn=LDAP-UX Proxy Users,cn=Groups, dc=hp,dc=com
uniquemember: cn=proxy1,cn=Users,dc=hp,dc=com
uniquemember: cn=proxy2,cn=Users,dc=hp,dc=com
```

```
# ldapsearch -D cn=proxy2,cn=Users,dc=hp,dc=com -w ldap1 -b "dc=hp,dc=com"
"cn=secure_profile"
dn: cn=secure_profile, ou=Secure Profiles, ou=ldap-ux, dc=hp, dc=com
objectclass: top
objectclass: DUAConfigprofile
preferredserverlist: 192.1.1.1:389
defaultsearchbase: ou=ldap-ux,dc=hp,dc=com
bindtimelimit: 5
authenticationmethod: simple
...
```

Now verify that anonymous users AND a user that is NOT a member of the LDAP-UX Administrators group can NOT modify a profile entry:

1st check for any ACI's on the entry itself:

```
# ldapsearch -h ldap1 -D cn=orcladmin -w ldapldap -b "ou=Secure Profiles,
ou=ldap-ux,dc=hp,dc=com" cn=secure_profile orclentrylevelaci orclaci
dn: cn=secure_profile, ou=Secure Profiles, ou=ldap-ux, dc=hp, dc=com
```

Now check for anonymous modification:

```
# ldapmodify -h ldap1
dn: cn=secure_profile, ou=Secure Profiles, ou=ldap-ux, dc=hp, dc=com
changetype: modify
replace: bindtimelimit
bindtimelimit: 999
```

modifying entry cn=secure_profile, ou=Secure Profiles, ou=ldap-ux, dc=hp, dc=com

ldap_modify: Insufficient access

Now check an authenticated, unauthorized, user modification:

```
# ldapmodify -h ldap1 -D cn=proxy2,cn=Users,dc=hp,dc=com -w ldap1
dn: cn=secure_profile, ou=Secure Profiles, ou=ldap-ux, dc=hp, dc=com
changetype: modify
replace: bindtimelimit
bindtimelimit: 999
```

modifying entry cn=secure_profile, ou=Secure Profiles, ou=ldap-ux, dc=hp, dc=com

ldap_modify: Insufficient access

Now for good measures test for authenticated, authorized user modification:

```
# ldapmodify -h ldap1 -D cn=Joe,cn=Users,dc=hp,dc=com -w ldap1234
dn: cn=secure_profile, ou=Secure Profiles, ou=ldap-ux, dc=hp, dc=com
changetype: modify
replace: bindtimelimit
bindtimelimit: 999
```

modifying entry cn=secure_profile, ou=Secure Profiles, ou=ldap-ux, dc=hp, dc=com

```
# ldapsearch -h ldap1 -D cn=orcladmin -w ldapldap -b "dc=hp,dc=com" "cn=secure_profile"
bindtimelimit
dn: cn=secure_profile, ou=Secure Profiles, ou=ldap-ux, dc=hp, dc=com
bindtimelimit: 999
```

NOTE: Although access to the configuration profile in the Directory has been restricted to proxy users only UNIX users logged in to an LDAP-UX client can view the local copy of the configuration profile:

```
$ id
uid=2004(ldapusr4) gid=20(users)
$ /opt/ldapux/config/display_profile_cache
```

LDAP-UX Client Services

Global Information from the Configuration Profile

```
host[:port]:      192.1.1.1:389
default search base: ou=ldap-ux,dc=hp,dc=com
auth:             simple
profilecachettl:  0 = infinite
follow referrals: enabled
search time limit: 0 = no limit
bind time limit:  5 seconds
credential level: anonymous
...
```

5.5.2 Posix Attributes

Similar to access control on configuration profiles it's important to make sure the right access to Posix attributes is set before deploying LDAP-UX in production. For example users should not be given write access to their own uidnumber. With the exception of the "userpassword" attribute it's not usually necessary to restrict access to Posix attributes of a user or group. Users of an LDAP-UX client will be able to view these attributes using the native Unix tools such as pwget or nsquery, however it may be desirable to restrict read access to non LDAP-UX users.

To restrict read access to Posix attributes, an ACI entry similar to the one in the configuration profile example above can be used. There difference between an ACI set on the configuration profile and one set on a user's entry is that the user may be allowed to change an attribute value on their own object. To give a user the ability to modify an attribute of their own object the user "self" can be given write access to the specific attributes:

```
orclaci: access to attr=(userpassword,loginshell)
        by * (none)
        by self (read, write, search, compare)
```

However with this ACI entry other users, including a proxy user, will not be able to view the user Posix attributes:

```
# ldapsearch -h ldap1 -b "ou=people,ou=ldap-ux,dc=hp,dc=com" "uid=ldapusr1"
ldap_search: Insufficient access
```

If no proxy user is being used with LDAP-UX read & search must be granted to anonymous users:

```
dn: ou=People,ou=ldap-ux,dc=hp,dc=com
orclaci: access to attr=(userpassword)
        by * (none) by self (read, write, search, compare)
orclaci: access to attr=(loginshell)
        by * (read, search, compare) by self (read, write, search, compare)
orclaci: access to attr=(*)
        by * (read, search, compare) by self (read, search, compare)
```

Anonymous users have read access to the users object, except the userpassword attribute:

```
# ldapsearch -h ldap1 -b "ou=people,ou=ldap-ux,dc=hp,dc=com" "uid=ldapusr1"
dn: uid=ldapusr1,ou=People,ou=ldap-ux,dc=hp,dc=com
uid: ldapusr1
cn: ldapusr1
objectclass: top
objectclass: account
objectclass: posixAccount
loginshell: /sbin/sh
uidnumber: 2001
gidnumber: 20
homedirectory: /home/ldapusr1
```

Using the ACL Check script to verify that a users attributes can not be modified by another user:

```
# posixAccount_acl_check "uid=ldapusr1,ou=People,ou=ldap-ux,dc=hp,dc=com" ldap1 \
        "uid=ldapusr2,ou=People,ou=ldap-ux,dc=hp,dc=com"
```

```
Trying to change attributes for:
uid=ldapusr2,ou=People,ou=ldap-ux,dc=hp,dc=com
with
uid=ldapusr1,ou=People,ou=ldap-ux,dc=hp,dc=com
```

```
Trying: homedirectory... Failed! Error: 50
Trying: uidnumber... Failed! Error: 50
Trying: loginshell... Failed! Error: 50
Trying: gidnumber... Failed! Error: 50
Trying: gecos... Failed! Error: 50
FAILED to change password!!
```

Now check what attributes the user can change on its own object:

```
# posixAccount_acl_check "uid=ldapusr1,ou=People,ou=ldap-ux,dc=hp,dc=com" ldap1
Trying to change attributes for:
uid=ldapusr1,ou=People,ou=ldap-ux,dc=hp,dc=com
with
uid=ldapusr1,ou=People,ou=ldap-ux,dc=hp,dc=com
```

```
Trying: homedirectory... Failed! Error: 50
Trying: uidnumber... Failed! Error: 50
Trying: loginshell... Succeeded!
Trying: gidnumber... Failed! Error: 50
Trying: gecos... Failed! Error: 50
Password changed to newPass1
```

When deploying LDAP-UX with a proxy user the same ACI entries can be used, allowing anonymous access to all but the userPassword attribute of the users object. If you wish to allow only proxy users (and the user itself) to view these attribute the above ACI must be modified to give the LDAP-UX Proxy Users group read access while removing all access to the anonymous users:

```
dn: ou=People,ou=ldap-ux,dc=hp,dc=com
orclaci: access to attr=(userpassword)
    by * (none)
    by self (read, write, search, compare)
orclaci: access to attr=(loginshell)
    by group="cn=LDAP-UX Proxy Users,cn=Groups,dc=hp,dc=com"
        (read, search,compare) by self (read, write, search, compare)
    by * (none)
orclaci: access to attr=(*)
    by group="cn=LDAP-UX Proxy Users,cn=Groups,dc=hp,dc=com"
        (read, search, compare) by self (read, search, compare)
    by * (none)
```

Anonymous access is denied:

```
# ldapsearch -h ldap1 -b "ou=people,ou=ldap-ux,dc=hp,dc=com" "uid=ldapusr1"
ldap_search: Insufficient access
```

but the user can see its own object:

```
# ldapsearch -h ldap1 -D "uid=ldapusr1,ou=People,ou=ldap-ux,dc=hp,dc=com" \
    -w newPass1 -b "dc=hp,dc=com" "uid=ldapusr1"
dn: uid=ldapusr1,ou=People,ou=ldap-ux,dc=hp,dc=com
userpassword: {MD4}aQJF6zyS2piyfUCZ/FWqGA==
uid: ldapusr1
cn: ldapusr1
objectclass: top
objectclass: account
objectclass: posixAccount
uidnumber: 2001
gidnumber: 20
homedirectory: /home/ldapusr1
loginshell: /usr/bin/badshell
```

but not another user's object:

```
# ldapsearch -h ldap1 -D "uid=ldapusr1,ou=People,ou=ldap-ux,dc=hp,dc=com" \
    -w newPass1 -b "dc=hp,dc=com" "uid=ldapusr2"
```

ldap_search: Insufficient access

and finally a proxy user can see the users object, except the userpassword attribute:

```
# ldapsearch -h ldap1 -D cn=proxy2,cn=Users,dc=hp,dc=com -w ldap1
-b "dc=hp,dc=com" "uid=ldapusr2"
dn: uid=ldapusr2,ou=People,ou=ldap-ux,dc=hp,dc=com
uid: ldapusr2
cn: ldapusr2
objectclass: top
objectclass: account
objectclass: posixAccount
loginshell: /sbin/sh
uidnumber: 2002
gidnumber: 20
homedirectory: /home/ldapusr2
```

```
# ldapsearch -h ldap1 -D cn=proxy2,cn=Users,dc=hp,dc=com -w ldap1
-b "dc=hp,dc=com" "uid=ldapusr2" userpassword
dn: uid=ldapusr2,ou=People,ou=ldap-ux,dc=hp,dc=com
```

Verify one user can not modify another user's object:

```
# posixAccount_acl_check "uid=ldapusr1,ou=People,ou=ldap-ux,dc=hp,dc=com" \
    newPass1 "uid=ldapusr2,ou=People,ou=ldap-ux,dc=hp,dc=com"
Trying to change attributes for:
uid=ldapusr2,ou=People,ou=ldap-ux,dc=hp,dc=com
with
uid=ldapusr1,ou=People,ou=ldap-ux,dc=hp,dc=com
```

```
Trying: homedirectory... Failed! Error: 50
Trying: uidnumber... Failed! Error: 50
Trying: loginshell... Failed! Error: 50
Trying: gidnumber... Failed! Error: 50
Trying: gecos... Failed! Error: 50
FAILED to change password!!
```

Verify a user can only change its own userpassword and loginshell attributes:

```
# posixAccount_acl_check "uid=ldapusr1,ou=People,ou=ldap-ux,dc=hp,dc=com" newPass1
Trying to change attributes for:
uid=ldapusr1,ou=People,ou=ldap-ux,dc=hp,dc=com
with
uid=ldapusr1,ou=People,ou=ldap-ux,dc=hp,dc=com
```

```
Trying: homedirectory... Failed! Error: 50
Trying: uidnumber... Failed! Error: 50
Trying: loginshell... Succeeded!
Trying: gidnumber... Failed! Error: 50
Trying: gecos... Failed! Error: 50
Password changed to newPass1
```


5.6 Password Management

Because LDAP-UX relies on the Directory Server to enforce the Password Policy the password policy configured by the Directory Administrator will apply to any HP-UX system configured with LDAP-UX. Basic testing with all password policy features of the Oracle Internet Directory has been successful. However, when a password modification or authentication fails the user is not informed why the operation failed. For example when a user attempts to change his password to one that does not meet the password policy of the Directory Server he will receive a generic error message:

```
# passwd
Old password:
New password:
Re-enter new password:
Failure - LDAP Processing Error
```

When using the `ldapmodify` command to attempt the same operation a more descriptive error message may be displayed:

```
# ldapmodify -h atc01 -D uid=ldapx1,ou=People,ou=ldap-ux,dc=hp,dc=com -w ldap1234
dn: uid=ldapx1,ou=People,ou=ldap-ux,dc=hp,dc=com
changetype: modify
replace: userpassword
userpassword: newpass

modifying entry uid=ldapx1,ou=People,ou=ldap-ux,dc=hp,dc=com
ldap_modify: DSA is unwilling to perform
ldap_modify: additional info: Password Policy Error :9004: GSL_PWDNUMERIC_EXCP :
Your Password must contain at least 3 numeric characters.
#
```

When creating new users it is important to create them with the `userpassword` attribute set. If this attribute is not set when the object is created the `pwdchangedtime` does not get set (this is the expected behavior). If a password is set on the object at a later time the `pwdchangedtime` does not get updated when it should (Oracle Bug: 2509657). Because password expiration features rely on this attribute being set the users password will never time out.

While it does not appear to be documented, password policy operational attributes are not set on users who have been created prior to implementing a password policy but have changed their passwords after the new password policy has been configured.

5.6.1 Oracle Internet Directory Password Policy

Oracle Internet Directory allows Administrators to apply multiple password policies to the Directory. Password policies can be set on a subset of users, all users as well as an individual user (or a combination of all 3). A password policy consists of 2 objects; the 1st is a

structural objectclass of type orclCommonAttributes, which identifies what the password policy will be applied to (defined in the orclcommonusersearchbase attribute):

```
objectclasses: ( 2.16.840.1.113894.7.2.1004 NAME 'orclCommonAttributes' SUP
                'orclContainer' STRUCTURAL
MAY ( orclCommonNicknameAttribute $ orclCommonApplicationGuidAttribute $
      orclCommonUserSearchBase $ orclCommonGroupSearchBase $
      orclCommonPasswordPolicy $ orclVersion ) )
```

The 2nd objectclass, pwdpolicy, contains the actual password policy to be applied.

```
objectclasses: ( 1.3.6.1.4.1.42.2.27.8.2.1 NAME 'pwdpolicy' SUP top STRUCTURAL
MUST ( cn )
MAY ( pwdMinAge $ pwdMaxAge $ pwdLockout $ pwdLockoutDuration $
      pwdMaxFailure $ pwdFailureCountInterval $ pwdExpireWarning $
      pwdCheckSyntax $ pwdSafeModify $ pwdMinLength $ pwdGraceLoginLimit $
      pwdMustChange $ orclpwdIllegalValues $ orclpwdAlphaNumeric $ orclpwdToggle $
      pwdInHistory $ pwdAllowUserChange ) )
```

See chapter 17 of the Oracle Internet Directory Administrators Guide for a detailed description of the available password policy attributes.

Example 1: LDIF file to create a Common Password Policy applied to all users under ou=People,ou=ldap-ux,dc=hp,dc=com

```
dn: cn=Common,ou=ldap-ux,dc=hp,dc=com
changetype: add
objectclass: top
objectclass: orclCommonAttributes
objectclass: orclContainer
objectclass: orclCommonAttributesV2
orcluserobjectclasses: orcluser
orcluserobjectclasses: orcluserv2
cn: Common
orclcommonusersearchbase: ou=People,ou=ldap-ux,dc=hp,dc=com
orclcommongroupsearchbase: ou=Group,ou=ldap-ux,dc=hp,dc=com
orclcommonnicknameattribute: cn
orclcommonapplicationguidattribute: orclGlobalID
orclversion: 90000
orclentrylevelaci: access to entry by * (browse,noadd,nodelete)
orclentrylevelaci: access to attr=(*) by group="cn=OracleDASConfiguration,
cn=Groups,cn=oraclecontext,dc=hp,dc=com" (read,write,search,compare) by *
(read,search,nowrite,nocompare)

dn: cn=PwdPolicyEntry,cn=Common,ou=ldap-ux,dc=hp,dc=com
changetype: add
objectclass: pwdpolicy
objectclass: top
cn: pwdpolicyentry
pwdmaxage: 5184000
pwdexpirewarning: 0
```

```

pwdmaxfailure: 10
pwdfailurecountinterval: 0
pwdlockout: 1
pwdlockoutduration: 86400
pwdchecksyntax: 1
pwdminlength: 5
orclpwdalphanumeric: 1
orclpwdtoggle: 1
orclpwdillegalvalues: passwd
orclpwdillegalvalues: password
pwdgracelogleinlimit: 1

```

Example 2: LDIF file to create a password policy that applies ONLY to the user
uid=ldapusr2,ou=People,ou=ldap-ux,dc=hp,dc=com

```

dn: cn=ldapusr2,ou=PasswdPolicies,ou=ldap-ux,dc=hp,dc=com
changetype: add
objectclass: top
objectclass: orclCommonAttributes
objectclass: orclContainer
objectclass: orclCommonAttributesV2
orcluserobjectclasses: orcluser
orcluserobjectclasses: orcluserv2
cn: ldapusr2
orclcommonusersearchbase: uid=ldapusr2,ou=People,ou=ldap-ux,dc=hp,dc=com
orclcommonnicknameattribute: cn
orclcommonapplicationguidattribute: orclGlobalID
orclversion: 90000
orclentrylevelaci: access to entry by * (browse,noadd,nodelete)
orclentrylevelaci: access to attr=(*) by group="cn=OracleDASConfiguration,
cn=Groups,cn=oraclecontext,dc=hp,dc=com" (read,write,search,compare) by *
(read,search,nowrite,nocompare)

```

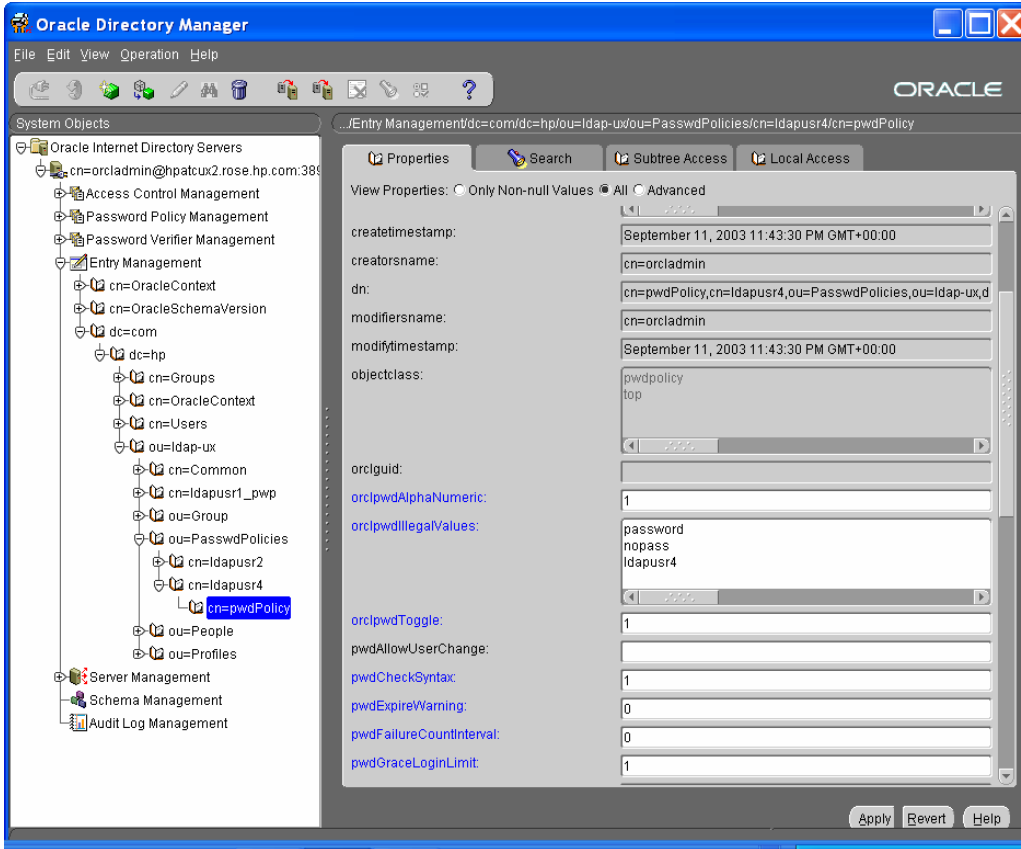
```

dn: cn=PwdPolicyEntry,cn=ldapusr2,ou=PasswdPolicies,ou=ldap-ux,dc=hp,dc=com
changetype: add
objectclass: pwdpolicy
objectclass: top
cn: pwdpolicyentry
pwdmaxage: 5184000
pwdexpirewarning: 0
pwdmaxfailure: 5
pwdfailurecountinterval: 0
pwdlockout: 1
pwdlockoutduration: 86400
pwdchecksyntax: 1
pwdminlength: 2
orclpwdalphanumeric: 1
orclpwdtoggle: 1
orclpwdillegalvalues: ldapusr2
orclpwdillegalvalues: password
pwdgracelogleinlimit: 1

```

When both of these password policies are implemented in the Directory Server the password policy specific to ldapusr2 will take precedence over the general password policy.

Password policies can also be created and modified using the Directory Management tool by 1st creating an orclCommonAttributes and then adding a pwdPolicy object



NOTE: The password policies are only applied to the userpassword attribute of the users object.

5.7 Migrating and Managing Users and Groups

The migration scripts that are delivered with LDAP-UX under the `/opt/ldapux/migrate` directory have been verified to work with Oracle Internet Directory (when using objectclasses and attributes defined in RFC2307). For more information regarding the migration scripts see the [Installing and Administering LDAP-UX Client Services](http://docs.hp.com) available at <http://docs.hp.com>.

5.8 Gotcha's

- Informative messages are not displayed when a password change fails due to a conflict with the OID password policy

5.9 Helpful Links

- LDAP Password Policy (IETF Draft): <http://www.ietf.org/internet-drafts/draft-behera-ldap-password-policy-06.txt>
- Oracle Internet Directory Administrators Guide:
http://otn.oracle.com/docs/products/oracle9i/doc_library/release2/network.920/a96574/toc.htm
- Oracle Internet Directory Developers Guide:
http://otn.oracle.com/docs/products/oracle9i/doc_library/release2/network.920/a96577/title.htm

6 OpenLDAP

The following pages address configuring [OpenLDAP](#) to store users and groups, while it is possible to store other NIS Maps in OpenLDAP this document does not discuss it. The following procedures and examples are taken while integrating LDAP-UX with OpenLDAP 2.1.3 on HP-UX 11.0.

To successfully integrate LDAP-UX with OpenLDAP perform the following steps (discussed in further detail below) in order:

1. Prepare Directory for LDAP-UX Integration
 - a. DUAConfig Profile Schema
 - b. RFC2307 Schema
 - c. Create a proxy user (Optional)
 - d. ACL
2. Create a configuration profile entry in the directory
3. Update local LDAP-UX configuration file(s)
4. Start ldapclientd
5. Configure `/etc/pam.conf` & `/etc/nsswitch.conf` (see `pam.ldap` and `nsswitch.ldap` for examples)
6. Test the LDAP-UX configuration (you'll need at least 1 POSIX user/group configured)
7. Migrate and/or Configure UNIX Users and Groups

6.1 Preparing OpenLDAP for LDAP-UX Integration

Prior to configuring LDAP-UX on your HP-UX system the following steps should be done in order to prepare the OpenLDAP Directory Server.

6.1.1 Adding the DUAConfig Profile Schema definition

OpenLDAP does not support online schema updates, therefore it is necessary to manually add the configuration profile schema definition file to OpenLDAP's configuration. A valid schema definition file for the DUAConfig profile can be found in Appendix C. To add the new schema definition file to OpenLDAP simply place the file containing the schema definition from Appendix C somewhere on the system running OpenLDAP. It is recommended to place the file with the schema files that are delivered with OpenLDAP: `<install dir>/etc/openldap/schema/`. Next add the file location to `slapd`'s configuration file, typically:

```
<install-dir>/etc/openldap/slapd.conf
```

Example:

```
$ cat /usr/local/etc/openldap/schema/duaconfig.schema
#
# This Schema file implements IETF Draft:
# "A Configuration Schema for LDAP Based Directory User Agents"
# draft-joslin-config-schema-06.txt
```

```

#
# URL: http://www.ietf.org/internet-drafts/draft-joslin-config-schema-06.txt
#

attributeType ( 1.3.6.1.4.1.11.1.3.1.1.3 NAME 'searchtimelimit' DESC 'Maximum time in
seconds a DUA should allow for a search to complete' EQUALITY integerMatch SYNTAX
1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE )

...

$ cat /usr/local/etc/openldap/slapd.conf
# $OpenLDAP: pkg/ldap/servers/slapd/slapd.conf,v 1.23 2002/02/02 05:23:12 kurt Exp $
#
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
include      /usr/local/etc/openldap/schema/core.schema
include      /usr/local/etc/openldap/schema/cosine.schema
include      /usr/local/etc/openldap/schema/inetorgperson.schema
include      /usr/local/etc/openldap/schema/duaconfig.schema
...

```

6.1.2 Extend the OpenLDAP Schema for RFC2307 (NIS/POSIX Attributes)

OpenLDAP delivers an RFC2307 schema definition file. In the default configuration file the RFC2307 schema definition file (nis.schema) is not included. To enable RFC 2307 add the nis.schema file, found in the <install dir>/etc/openldap/schema directory, to the OpenLDAP Servers configuration file:

```

$ cat /usr/local/etc/openldap/slapd.conf
# $OpenLDAP: pkg/ldap/servers/slapd/slapd.conf,v 1.23 2002/02/02 05:23:12 kurt Exp $
#
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
include      /usr/local/etc/openldap/schema/core.schema
include      /usr/local/etc/openldap/schema/cosine.schema
include      /usr/local/etc/openldap/schema/inetorgperson.schema
include      /usr/local/etc/openldap/schema/duaconfig.schema
include      /usr/local/etc/openldap/schema/nis.schema

```

6.1.3 Configure a Proxy Server (*Optional*)

In a default OpenLDAP installation/configuration a proxy user is NOT required for LDAP-UX to function properly. It is possible to modify the access control such that a proxy user is required for LDAP-UX. For more details on configuring proxy user access see the section title *“Proxy Users”* below.

6.1.4 Access Control (ACL's)

While it is not required to configure ACL's during the preparation stage this may be a good time to set ACL's on Posix user and group objects. See the section titled Access Control for more details.

6.2 Profile Schema

Because the setup tool is not being used for configuration the profile schema entry in the Directory must be created and added manually. Appendix D provides a sample LDIF file that can be used as a template.

It's important to make sure that when the profile is added to the Directory the proper access control is placed on the entry. The profile entry should either be readable by everyone or by the proxy user (or proxy user group if using multiple proxy users). The entry should only be writable by an administrator principle. The following example details the most common attributes of a profile configuration entry:

```
# DN of the Profile Entry
dn: cn=ldapuxprofile2, ou=profiles,ou=ldap-ux,dc=acme,dc=com

# Object Classes
objectClass: top
objectClass: duaconfigprofile

# Common Name for the profile
cn: ldapuxprofile2

# The list of Directory Servers and ports for LDAP-UX to search.
# NOTE: The configuration profile does NOT need to reside on the same Directory Server as
#       the users and groups do.
preferredserverlist: 192.1.1.1:389 192.1.1.2:444

# Entry within the Directory LDAP-UX will begin to search for Users and Groups
defaultsearchbase: ou=ldap-ux,dc=acme,dc=com

# The maximum time in seconds the client should wait for a directory search to complete
searchtimelimit: 45

# The maximum time in seconds the client should wait to connect to the server
bindtimelimit: 5

# Authentication method for users to bind/authenticate to the server
authenticationmethod: simple

# How long before the configuration profile should be refreshed from the Directory (in
# seconds)
profilettl: 86400

# How LDAP-UX will bind to the Directory (anonymous OR proxy OR proxy anonymous)
credentiallevel: proxy anonymous

# Map one attribute to another (multi-valued attribute)
attributemap: passwd:userpassword=*NULL*
attributemap: shadow:userpassword=*NULL*
```

6.3 Proxy User

If a proxy user will be used with LDAP-UX it must be created in the Directory and configured on the LDAP-UX system **before** enabling LDAP-UX. The proxy user can be created in the Directory just as any other user would be, although it should not have Posix attributes as this user should not be used to login to the HP-UX system. Once the user is created in the Directory the HP-UX system must be configured with the proxy users DN and password (stored encrypted).

To configure the proxy user into the HP-UX system use the `ldap_proxy_config` command. There are several ways to configure the Proxy user, see the man page for details. To configure the Proxy user with one command execute:

```
ldap_proxy_config -d <Proxy User DN> -c <Password>
```

To view the DN of the Proxy user that is configured in the system use the '-p' option:

```
ldap_proxy_config -p
```

To verify the password and DN configured use the `-v` option:

```
ldap_proxy_config -v
```

NOTE: The `ldap_proxy_config -v` command cannot verify the Proxy DN and password until LDAP-UX is completely configured (See section 6.4).

```
# /opt/ldapux/config/ldap_proxy_config \  
-d "cn=proxy-hpatcdl,ou=ProxyUsers,ou=ldap-ux,dc=acme,dc=com" -c ldap
```

```
# /opt/ldapux/config/ldap_proxy_config -p  
PROXY DN: cn=proxy-hpatcdl,ou=ProxyUsers,ou=ldap-ux,dc=acme,dc=com
```

6.4 Configuring LDAP-UX client

Now that the LDAP-UX profile is created in the Directory and a proxy user has been created and configured on the local LDAP-UX host it's time to configure the LDAP-UX client. The LDAP-UX client configuration consists of 3 files, `ldapux_client.conf` and `ldapux_profile.bin` and `ldapux_profile.ldif`. The `ldapux_client.conf` file is a "bootstrap" file. This file only contains enough information to tell the LDAP-UX client where to download its configuration profile from. Once the LDAP-UX system has downloaded the configuration profile it is stored in 2 files `ldapux_profile.bin` (binary format) and `ldapux_profile.ldif` (ldif format).

The 1st step is to update the `/etc/opt/ldapux/ldapux_client.conf` file. At the bottom of the file there is a Profile section, denoted by the "[profile]" string. In this section add, or update, the following lines:

```
Service: NSS
LDAP_HOSTPORT=<Directory IP Address>:<Port>
PROFILE_ENTRY_DN=<Profile DN> (created in section 6.2)
PROGRAM="/opt/ldapux/config/create_profile_cache"
```

Example:

```
Service: NSS
LDAP_HOSTPORT="192.1.1.1:389"
PROFILE_ENTRY_DN="cn=ldapux-profile2,ou=profiles,ou=ldap-ux,dc=acme,dc=com"
PROGRAM="/opt/ldapux/config/create_profile_cache"
```

Now that the "boot" file is configured and the Proxy user is configured the profile can now be downloaded using the `get_profile_entry` command:

```
# /opt/ldapux/config/get_profile_entry -s nss
```

If the proxy user is configured incorrectly, or the proxy user does not have read access to the profile entry, this error will be displayed when attempting to download the profile:

```
# /opt/ldapux/config/get_profile_entry -s nss
```

```
PFMERR 5: Downloading profile entry failed!
```

Executing `ldapsearch` on the profile entry while binding to the directory with the proxy users credential may provide additional information on the problem:

```
# ldapsearch -b "dc=acme,dc=com" \
-D "cn=proxy-hpatcdl,ou=ProxyUsers,ou=ldap-ux,dc=acme,dc=com" \
-w ldapldap cn=ldapux-profile2
ldap_bind: Invalid credentials
```

After `get_profile_entry` is executed the `ldapux_profile.bin` and `ldapux_profile.ldif` files should be created in the `/etc/opt/ldapux` directory. The `ldapux_profile.ldif` should look similar to the `ldif` file used to create the entry in the Directory.

To view, and verify, the profile configuration use the `display_profile_cache` command:

```
# /opt/ldapux/config/display_profile_cache
```

LDAP-UX Client Services

Global Information from the Configuration Profile

```
host[:port]:      ldap1.hp.com
default search base: dc=acme,dc=com
auth:             simple
profilecachettl:  0 = infinite
follow referrals: enabled
search time limit: 0 = no limit
bind time limit:  5 seconds
credential level: proxy
```

...

6.5 Enabling ldapclientd

Prior to LDAP-UX version 3.2 the ldapclientd daemon is not required to be running in order for LDAP-UX to function correctly but it does serve several important functions, in particular:

- Improves performance by caching entries
- Automatically downloading configuration profiles (based on the profile TTL)

In order for ldapclientd to start up at system boot time the `/etc/opt/ldapux/ldapclientd.conf` file must be modified by setting “enabled=yes” under the [StartOnBoot] section:

```
[StartOnBoot]
enable=yes
```

To start the ldapclientd use the startup script `/sbin/init.d/ldapclientd.rc`:

```
# /sbin/init.d/ldapclientd.rc start
ldapclientd started with <0>
```

For more information on the features and configurable parameters of ldapclientd see `man 1M ldapclientd`.

6.6 Enabling LDAP-UX

At this point the HP-UX system is now configured for LDAP-UX, but it has not been enable to use LDAP for users or groups.

To enable the HP-UX system to use LDAP-UX to retrieve posix users and groups the Name Service Switch must be configured. The Name Service Switch is configured in the file `/etc/nsswitch.conf`, for details on `nsswitch.conf` see `man 4 switch`. A sample `nsswitch` configuration for LDAP-UX can be found in the file `/etc/nsswitch.ldap`. The following basic configuration can be used to search local files 1st and then LDAP if the user or group is not found:

```
passwd:      files ldap
group:      files ldap
```

This is a basic configuration that only shows users (`passwd`) and groups being retrieved through LDAP-UX. It is possible, and likely that a more complex configuration will be required to meet you're needs.

After configuring the Name Service Switch use the `nsquery` and/or `pwget` commands to verify the configuration (this will require that a user and/or group in the directory with Posix attributes already exist):

```
# /usr/contrib/bin/nsquery passwd ldapusr4
```

Using "files ldap" for the passwd policy.

```
Searching /etc/passwd for ldapusr4
ldapusr4 was NOTFOUND
```

Switch configuration: Allows fallback

```
Searching ldap for ldapusr4
User name: ldapusr4
User Id: 1004
Group Id: 20
Gecos:
Home Directory: /home/ldapusr4
Shell: /usr/bin/sh
```

Switch configuration: Terminates Search

```
# pwget -u 1004
ldapusr4*:1004:20::/home/ldapusr4:/usr/bin/sh
# pwget -n ldapusr4
ldapusr4*:1004:20::/home/ldapusr4:/usr/bin/sh
```

After verifying that user and groups can be retrieved using LDAP-UX its time to enable user authentication through LDAP-UX. Authentication on HP-UX is configured through PAM (Pluggable Authentication Module). The PAM configuration file is `/etc.pam.conf`. The LDAP-UX product also supplies a sample PAM configuration file already configured to use LDAP-UX for authentication. This sample configuration file is `/etc/pam.ldap`.

The following example shows how to configure `pam.conf` for the **“login”** service. There are several services that can be configured, see `man pam.conf` for more details.

```
#
# Try PAM_UNIX 1st for local (root) users. If that fails try PAM_LDAP, trying the original
# password entered by the user.
#
login  auth          sufficient  /usr/lib/security/libpam_unix.1
login  auth          required    /usr/lib/security/libpam_ldap.1 try_first_pass
login  account       sufficient  /usr/lib/security/libpam_unix.1
login  account       required    /usr/lib/security/libpam_ldap.1
login  session       sufficient  /usr/lib/security/libpam_unix.1
login  session       required    /usr/lib/security/libpam_ldap.1
login  password      sufficient  /usr/lib/security/libpam_unix.1
login  password      required    /usr/lib/security/libpam_ldap.1 try_first_pass
```

The best way to test out PAM authentication is to login to the system with a user/password stored in the Directory. Check `/var/adm/syslog/syslog.log` for errors.

6.7 POSIX Attributes (RFC 2307)

The NIS/Posix/RFC2307 schema definition file is delivered with OpenLDAP. Enabling support for RFC2307 requires that the schema definition file be included in the OpenLDAP server's configuration file as described in section 6.1.2.

6.8 Access Control

6.8.1 Configuration Profile

A mentioned earlier access must be set on the configuration profile such that the LDAP-UX system can read/download the entry. The default ACL entry, delivered with OpenLDAP's slapd.conf file, allows:

- read access of root DSE
- self write access
- authenticated users read access
- anonymous users to authenticate

If a proxy user is not configured the LDAP-UX system will not be able to download the profile. If no proxy users will be used an ACL must be set on the profile entry giving anonymous read access. For example:

```
access to dn="cn=ldapux-profile2,ou=profiles,ou=ldap-ux,dc=acme,dc=com"
    by anonymous read
```

6.8.2 Posix Attributes

Again setting ACL's on Posix attributes will depend on whether or not a proxy user is being used. If no proxy user is being used the LDAP-UX client will be searching of user and group entries with an anonymous bind. If the default ACL's are set LDAP-UX will not retrieve any entries from the Directory. There are several ACL options to allow an anonymous user to read Posix information. The follow is just one simple example that grants anonymous users read access to all entries in the ou=ldap-ux,dc=acme,dc=com subtree except the userPassword attribute:

```
access to attr=userPassword
    by self write
    by anonymous auth
    by * none
```

```
access to dn.children="ou=ldap-ux,dc=acme,dc=com"
    by self write
    by anonymous read
```

There is a problem with the above ACL; authenticated users have self write set on their entry which means they can modify attributes such as uidnumber, gidnumber. Using the posixAccount_acl_check script verifies this:

```
# posixAccount_acl_check "uid=ldapusr4,ou=People,ou=ldap-ux,dc=acme,dc=com" newPass
```

Trying to change attributes for:

```
uid=ldapusr4,ou=People,ou=ldap-ux,dc=acme,dc=com
```

with

```
uid=ldapusr4,ou=People,ou=ldap-ux,dc=acme,dc=com
```

Trying: homedirectory... Succeeded!

Trying: uidnumber... Succeeded!

Trying: loginshell... Succeeded!

Trying: gidnumber... Succeeded!

Trying: gecos... Succeeded!

Password changed to newPass1

Allowing a user to change their uid, uidnumber or gidnumber is **BAD!**

Add the following in the middle of the ACI entry above will restrict users from being able to change their uidnumber, gidnumber or uid:

```
access to dn.children="ou=ldap-ux,dc=acme,dc=com" attr=uidnumber,gidnumber,uid
by * read
```

Now verify the ACL is providing the access control expected:

```
# posixAccount_acl_check "uid=ldapusr4,ou=People,ou=ldap-ux,dc=acme,dc=com" oldPass
```

Trying to change attributes for:

```
uid=ldapusr4,ou=People,ou=ldap-ux,dc=acme,dc=com
```

with

```
uid=ldapusr4,ou=People,ou=ldap-ux,dc=acme,dc=com
```

Trying: homedirectory... Succeeded!

Trying: uidnumber... Failed! Error: 50

Trying: loginshell... Succeeded!

Trying: gidnumber... Failed! Error: 50

Trying: gecos... Succeeded!

Password changed to newPass1

6.9 Password Management

OpenLDAP does not currently support a password policy such as password expiration time, password length, etc. Therefore there is nothing for LDAP-UX to enforce. The password-changing feature of PAM_LDAP is supported with OpenLDAP. This means that HP-UX users may use the `passwd` command to change their passwords on the OpenLDAP Server.

6.10 Migrating and Managing Users and Groups

The migration scripts that are delivered with LDAP-UX under the `/opt/ldapux/migrate` directory have been verified to work with OpenLDAP without modifications. For more information regarding the migration scripts see the [Installing and Administering LDAP-UX Client Services](http://docs.hp.com) available at <http://docs.hp.com>.

7 Scripts

Below are a number of sample (unsupported) scripts to help with deploying and verifying LDAP-UX configurations. These scripts are provided “as is”, so use at your own risk.

7.1 `posixAccount_acl_check`

This script is used to attempt to modify various `posixAccount` attributes of a user. The script takes 3 values:

1. User DN to authenticate as
2. Password for the user to authenticate as
3. User DN to attempt to authenticate as (Optional, defaults to the User DN in parm 1)

Sample run:

```
# posixAccount_acl_check "uid=ldapusr2,ou=People,ou=ldap-ux,dc=hp,dc=com" ldap
Trying to change attributes for:
  uid=ldapusr2,ou=People,ou=ldap-ux,dc=hp,dc=com
with
  uid=ldapusr2,ou=People,ou=ldap-ux,dc=hp,dc=com

Trying: homedirectory... Failed! Error: 49
Trying: uidnumber... Failed! Error: 49
Trying: userpassword... Succeeded!
Trying: loginshell... Failed! Error: 49
Trying: gidnumber... Failed! Error: 49
Trying: gecos... Failed! Error: 49

# posixAccount_acl_check "uid=ldapusr2,ou=People,ou=ldap-ux,dc=hp,dc=com" newPass1 \
  "uid=ldapusr1,ou=People,ou=ldap-ux,dc=hp,dc=com"
Trying to change attributes for:
  uid=ldapusr1,ou=People,ou=ldap-ux,dc=hp,dc=com
with
  uid=ldapusr2,ou=People,ou=ldap-ux,dc=hp,dc=com

Trying: homedirectory... Failed! Error: 50
Trying: uidnumber... Failed! Error: 50
Trying: userpassword... Failed! Error: 50
Trying: loginshell... Failed! Error: 50
Trying: gidnumber... Failed! Error: 50
Trying: gecos... Failed! Error: 50
```

Script Source:

```
#!/bin/ksh

export SHLIB_PATH=/opt/ldapux/lib
```

```

if [[ $# -lt 2 ]]
then
  echo "Usage: $0 <User DN> <User Password> [<Other User DN>]"
  echo " <User DN> : Users DN to authenticate as"
  echo " <User Password> : Users Password"
  echo " <Other User DN> : Users DN to modify (optional, defaults to User DN) "

  exit 99
fi

BIND_DN=$1
BIND_CRED=$2

if [[ $# -eq 2 ]]
then
  USER_ENTRY=$BIND_DN
else
  USER_ENTRY=$3
fi

HOST=`grep ^preferredserverlist /etc/opt/ldapux/ldapux_profile.ldif | awk '{print $2}'`

echo "Trying to change attributes for: \n $USER_ENTRY"
echo " with\n $BIND_DN\n"

for val in "homedirectory:/home/bad" "uidnumber:0" "loginshell:/usr/bin/badshell"
"gidnumber:0" "gecos:parm1,parm2,pamr3,pamr4"
do
  attr=`echo $val | awk -F: '{print $1}'`
  new_val=`echo $val | awk -F: '{print $2}'`
  echo "Trying: $attr... \c"

/opt/ldapux/bin/ldapmodify -h $HOST -D $BIND_DN \
-w $BIND_CRED >/dev/null 2>&1 <<-EOF
dn: $USER_ENTRY
changetype: modify
replace: $attr
$attr: $new_val

EOF
ret=$?
if [[ $ret -eq 0 ]]
then
  echo " Succeeded!"
else
  echo "Failed! Error: $ret"
fi

done

/opt/ldapux/bin/ldapmodify -h $HOST -D $BIND_DN -w $BIND_CRED >/dev/null 2>&1 <
<-EOF

```

```
dn: $USER_ENTRY
changetype: modify
replace: userPassword
userPassword: newPass1
```

```
EOF
```

```
ret=$?
if [[ $ret -eq 0 ]]
then
  echo "Password changed to newPass1 "
else
  echo "FAILED to change password!!"
fi
```

7.2 posixGroup_acl_check

This script is used to attempt to modify various posixGroup attributes of a user. The script takes 3 values:

1. Group DN to attempt to modify
2. User DN to authenticate as (Optional, defaults to anonymous bind)
3. Password for the user to authenticate as (Optional)

Sample run:

```
# posixGroup_acl_check "cn=staff,ou=Group,ou=ldap-ux,dc=hp,dc=com"
Trying to modify Group attributes for:
  cn=staff,ou=Group,ou=ldap-ux,dc=hp,dc=com
with
  Anonymous credentials

Trying: gidnumber... Failed! Error: 50
Trying: userpassword... Failed! Error: 50
Trying: memberuid... Failed! Error: 50

# posixGroup_acl_check "cn=staff,ou=Group,ou=ldap-ux,dc=hp,dc=com" \
  "uid=ldapusr2,ou=People,ou=ldap-ux,dc=hp,dc=com" newPass1
Trying to modify Group attributes for:
  cn=staff,ou=Group,ou=ldap-ux,dc=hp,dc=com
with
  uid=ldapusr2,ou=People,ou=ldap-ux,dc=hp,dc=com

Trying: gidnumber... Failed! Error: 50
Trying: userpassword... Failed! Error: 50
Trying: memberuid... Failed! Error: 50

# Using the Directory Administrator account:

# posixGroup_acl_check "cn=staff,ou=Group,ou=ldap-ux,dc=hp,dc=com" \
  "cn=orcladmin" welcome
```



```
Trying to modify Group attributes for:
  cn=staff,ou=Group,ou=ldap-ux,dc=hp,dc=com
with
  cn=orcladmin
```

```
Trying: gidnumber... Succeeded!
Trying: userpassword... Succeeded!
Trying: memberuid... Succeeded!
```

Script Source:

```
#!/bin/ksh

export SHLIB_PATH=/opt/ldapux/lib

if [[ $# -lt 1 ]]
then
  echo "Usage: $0 <Group DN> [<User DN> <User Password>]"
  echo " <Group DN> : Group DN to modify"
  echo " <User DN> : Users DN to authenticate as (optional, default to anonymous)"
  echo " <User Password> : Users Password (optional)"
  exit 99
fi

GROUP=$1

if [[ $# -eq 3 ]]
then
  BIND_DN=$2
  BIND_CRED=$3
fi

HOST=`grep ^preferredserverlist /etc/opt/ldapux/ldapux_profile.ldif | awk '{pri
nt $2}'`

echo "Trying to modify Group attributes for: \n $GROUP"
echo " with"

if [[ $# -lt 3 ]]
then
  echo " Anonymous credentials \n"
else
  echo " $BIND_DN \n"
fi

for val in "gidnumber:0" "userpassword:newPass1" "memberuid:user$$"
do
  attr=`echo $val | awk -F: '{print $1}'`
  new_val=`echo $val | awk -F: '{print $2}'`
  echo "Trying: $attr... \c"

if [[ $# -lt 3 ]]
```

```
then
/opt/ldapux/bin/ldapmodify -h $HOST >/dev/null 2>&1 <<-EOF
dn: $GROUP
changetype: modify
replace: $attr
$attr: $new_val

EOF
else
/opt/ldapux/bin/ldapmodify -h $HOST -D $BIND_DN -w $BIND_CRED >/dev/null 2>&1 <<
-EOF
dn: $GROUP
changetype: modify
replace: $attr
$attr: $new_val

EOF
fi
ret=$?
if [[ $ret -eq 0 ]]
then
echo " Succeeded!"
else
echo "Failed! Error: $ret"
fi
done
```

Appendix A (*eDirectory RFC2307 Schema Files*)

rfc2307-usergroup.sch:

```

RFC2307UserGroupSchemaExtensions DEFINITIONS ::=
BEGIN

-- An integer uniquely identifying a user in an administrative domain
"uidNumber" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_INTEGER,
    Flags              { DS_SINGLE_VALUED_ATTR },
    ASN1ObjID { 1 3 6 1 1 1 1 0 }
}

-- An integer uniquely identifying a group in an administrative domain
"gidNumber" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_INTEGER,
    Flags              { DS_SINGLE_VALUED_ATTR },
    ASN1ObjID { 1 3 6 1 1 1 1 1 }
}

-- The GECOS field; the common name
"gecos" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_CI_STRING,
    Flags              { DS_STRING_ATTR, DS_SINGLE_VALUED_ATTR },
    ASN1ObjID { 1 3 6 1 1 1 1 2 }
}

-- The absolute path to the home directory
"homeDirectory" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_CE_STRING,
    Flags              { DS_SINGLE_VALUED_ATTR },
    ASN1ObjID { 1 3 6 1 1 1 1 3 }
}

-- The path to the login shell
"loginShell" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_CE_STRING,
    Flags              { DS_SINGLE_VALUED_ATTR },
    ASN1ObjID { 1 3 6 1 1 1 1 4 }
}

```

```
"shadowLastChange" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_INTEGER,
    Flags              { DS_SINGLE_VALUED_ATTR },
    ASN1ObjID { 1 3 6 1 1 1 1 5 }
}
```

```
"shadowMin" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_INTEGER,
    Flags              { DS_SINGLE_VALUED_ATTR },
    ASN1ObjID { 1 3 6 1 1 1 1 6 }
}
```

```
"shadowMax" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_INTEGER,
    Flags              { DS_SINGLE_VALUED_ATTR },
    ASN1ObjID { 1 3 6 1 1 1 1 7 }
}
```

```
"shadowWarning" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_INTEGER,
    Flags              { DS_SINGLE_VALUED_ATTR },
    ASN1ObjID { 1 3 6 1 1 1 1 8 }
}
```

```
"shadowInactive" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_INTEGER,
    Flags              { DS_SINGLE_VALUED_ATTR },
    ASN1ObjID { 1 3 6 1 1 1 1 9 }
}
```

```
"shadowExpire" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_INTEGER,
    Flags              { DS_SINGLE_VALUED_ATTR },
    ASN1ObjID { 1 3 6 1 1 1 1 10 }
}
```

```
"shadowFlag" ATTRIBUTE ::=
{
    Operation          ADD,
```

```

SyntaxID          SYN_INTEGER,
Flags             { DS_SINGLE_VALUED_ATTR },
ASN1ObjID { 1 3 6 1 1 1 1 11 }
}

-- The uids of the users which belong to the group
"memberUid" ATTRIBUTE ::=
{
    Operation      ADD,
    SyntaxID       SYN_CE_STRING,
    ASN1ObjID { 1 3 6 1 1 1 1 12 }
}

-- Abstraction of an account with POSIX attributes
"posixAccount" OBJECT-CLASS ::=
{
    Operation      ADD,
    Flags          {DS_AUXILIARY_CLASS},
    SubClassOf     {"TOP"},
    MustContain    { "CN" },
    MustContain    { "uniqueID" },
    MustContain    { "uidNumber" },
    MustContain    { "gidNumber" },
    MustContain    { "homeDirectory" },
    MayContain     { "loginShell" },
    MayContain     { "gecos" },
    MayContain     { "description" },
    ASN1ObjID { 1 3 6 1 1 1 2 0 }
}

"shadowAccount" OBJECT-CLASS ::=
{
    Operation      ADD,
    Flags          {DS_AUXILIARY_CLASS},
    SubClassOf     {"TOP"},
    MustContain    { "uniqueID" },
    MayContain     { "shadowLastChange" },
    MayContain     { "shadowMin" },
    MayContain     { "shadowMax" },
    MayContain     { "shadowWarning" },
    MayContain     { "shadowInactive" },
    MayContain     { "shadowExpire" },
    MayContain     { "shadowFlag" },
    MayContain     { "description" },
    ASN1ObjID { 1 3 6 1 1 1 2 1 }
}

-- Abstraction of a group of accounts
"posixGroup" OBJECT-CLASS ::=
{
    Operation      ADD,
    Flags          {DS_AUXILIARY_CLASS},
    SubClassOf     {"TOP"},

```

```

    MustContain      {      "cn" },
    MustContain      {      "gidNumber" },
    MayContain       {      "memberUid" },
    MayContain       {      "description"},
    ASN1ObjID { 1 3 6 1 1 1 2 2 }
}

END

```

rfc2307-nis.sch:

```

RFC2307NisSchemaExtensions DEFINITIONS ::=
BEGIN

```

```

"memberNisNetGroup" ATTRIBUTE ::=

```

```

{
    Operation          ADD,
    SyntaxID           SYN_CE_STRING,
    Flags              { DS_SYNC_IMMEDIATE},
    ASN1ObjID          {1 3 6 1 1 1 13}
}

```

```

-- NetGroup Triple

```

```

-- rfc2307 defines a new syntax nisNetgroupTripleSyntax, which we cant define

```

```

-- in NDS. So, using the bit string syntax instead

```

```

"nisNetGroupTriple" ATTRIBUTE ::=

```

```

{
    Operation          ADD,
    SyntaxID           SYN_OCTET_STRING,
    Flags              { DS_SYNC_IMMEDIATE},
    ASN1ObjID          {1 3 6 1 1 1 14}
}

```

```

"ipServicePort" ATTRIBUTE ::=

```

```

{
    Operation          ADD,
    SyntaxID           SYN_INTEGER,
    Flags              { DS_SINGLE_VALUED_ATTR, DS_SYNC_IMMEDIATE},
    ASN1ObjID          {1 3 6 1 1 1 15}
}

```

```

"ipServiceProtocol" ATTRIBUTE ::=

```

```

{
    Operation          ADD,
    SyntaxID           SYN_CL_STRING,
    Flags              { DS_SIZED_ATTR, DS_SYNC_IMMEDIATE},
    LowerBound         1,
    UpperBound         32768,
    ASN1ObjID          {1 3 6 1 1 1 16}
}

```

```

"ipProtocolNumber" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_INTEGER,
    Flags              { DS_SINGLE_VALUED_ATTR, DS_SYNC_IMMEDIATE},
    ASN1ObjID          {1 3 6 1 1 1 1 17}
}

"oncRpcNumber" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_INTEGER,
    Flags              { DS_SINGLE_VALUED_ATTR, DS_SYNC_IMMEDIATE},
    ASN1ObjID          {1 3 6 1 1 1 1 18}
}

-- IP address as a dotted decimal, eg 192.168.1.1, omitting leading zeros
"ipHostNumber" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_CL_STRING,
    Flags              { DS_SIZED_ATTR, DS_SYNC_IMMEDIATE},
    LowerBound         1,
    UpperBound         128,
    ASN1ObjID          {1 3 6 1 1 1 1 19}
}

-- IP network as a dotted decimal, eg. 192.168, omitting leading zeros
"ipNetworkNumber" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_CL_STRING,
    Flags              { DS_SIZED_ATTR, DS_SINGLE_VALUED_ATTR,
DS_SYNC_IMMEDIATE},
    LowerBound         1,
    UpperBound         128,
    ASN1ObjID          {1 3 6 1 1 1 1 20}
}

-- IP netmask as a dotted decimal, eg. 255.255.255.0, omitting leading zeros
"ipNetmaskNumber" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_CL_STRING,
    Flags              { DS_SIZED_ATTR, DS_SINGLE_VALUED_ATTR,
DS_SYNC_IMMEDIATE},
    LowerBound         1,
    UpperBound         128,
    ASN1ObjID          {1 3 6 1 1 1 1 21}
}

-- MAC address in maximal, colon separated hex notation, eg. 00:00:92:90:ee:e2
"macAddress" ATTRIBUTE ::=

```

```

{
    Operation          ADD,
    SyntaxID           SYN_CI_STRING,
    Flags              { DS_SIZED_ATTR, DS_SYNC_IMMEDIATE},
    LowerBound         1,
    UpperBound         128,
    ASN1ObjID          {1 3 6 1 1 1 1 22}
}

-- rpc.bootparamd parameter
-- rfc2307 defines a new syntax bootParameterSyntax, which we cant define in
-- NDS. So, using the bit string syntax instead
"bootParameter" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_OCTET_STRING,
    Flags              { DS_SYNC_IMMEDIATE},
    ASN1ObjID          {1 3 6 1 1 1 1 23}
}

-- Boot image name
"bootFile" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_CE_STRING,
    Flags              { DS_SYNC_IMMEDIATE},
    ASN1ObjID          {1 3 6 1 1 1 1 24}
}

"nisMapName" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_CI_STRING,
    Flags              { DS_SIZED_ATTR, DS_SYNC_IMMEDIATE },
    LowerBound         1,
    UpperBound         64,
    ASN1ObjID          {1 3 6 1 1 1 1 26}
}

"nisMapEntry" ATTRIBUTE ::=
{
    Operation          ADD,
    SyntaxID           SYN_CE_STRING,
    Flags              { DS_SIZED_ATTR, DS_SINGLE_VALUED_ATTR,
    DS_SYNC_IMMEDIATE},
    LowerBound         1,
    UpperBound         1024,
    ASN1ObjID          {1 3 6 1 1 1 1 27}
}

-- Adding ATTRIBUTE manager.
-- This is present in the base schema of newer versions of NDS.
"manager" ATTRIBUTE ::=

```



```

{
    Operation      ADD,
    SyntaxID      SYN_DIST_NAME,
    Flags         { DS_SYNC_IMMEDIATE },
    ASN1ObjID     {0 9 2342 19200300 100 1 10}
}

-- Abstraction an Internet Protocol service. Maps an IP port and protocol
-- (such as tcp or udp) to one or more names; the distinguished value of the
-- cn attribute denotes the service's canonical name
"ipService" OBJECT-CLASS ::=
{
    Operation     ADD,
    Flags         { DS_EFFECTIVE_CLASS },
    SubClassOf    { "TOP" },
    ContainedBy   { "Organization", "Organizational Unit" },
    NamedBy       { "CN", "ipServiceProtocol" },
    MustContain   { "CN", "ipServicePort", "ipServiceProtocol" },
    MayContain    { "Description" },
    ASN1ObjID     {1 3 6 1 1 1 2 3}
}

-- Abstraction of an IP protocol. Maps a protocol number to one or more names.
-- The distinguished value of the cn attribute denotes the protocol's
-- canonical name
"ipProtocol" OBJECT-CLASS ::=
{
    Operation     ADD,
    Flags         { DS_EFFECTIVE_CLASS },
    SubClassOf    { "TOP" },
    ContainedBy   { "Organization", "Organizational Unit" },
    NamedBy       { "CN" },
    MustContain   { "CN", "ipProtocolNumber" },
    MayContain    { "Description" },
    ASN1ObjID     {1 3 6 1 1 1 2 4}
}

-- Abstraction of an Open Network Computing (ONC) [RFC1057] Remote Procedure
-- Call (RPC) binding. This class maps an ONC RPC number to a name. The
-- distinguished value of the cn attribute denotes the RPC service's
-- canonical name
"oncRpc" OBJECT-CLASS ::=
{
    Operation     ADD,
    Flags         { DS_EFFECTIVE_CLASS },
    SubClassOf    { "TOP" },
    ContainedBy   { "Organization", "Organizational Unit" },
    NamedBy       { "CN" },
    MustContain   { "CN", "oncRpcNumber" },
    MayContain    { "Description" },
    ASN1ObjID     {1 3 6 1 1 1 2 5}
}

```

```

-- Abstraction of a host, an IP device. The distinguished value of the cn
-- attribute denotes the host's canonical name.
-- According to RFC 2307 this is an Auxiliary class and device should be used
-- as Structural class. Currently we won't be able to do that so we are making
-- this as structural itself.
"ipHost" OBJECT-CLASS ::=
{
    Operation  ADD,
    Flags      { DS_EFFECTIVE_CLASS },
    SubClassOf { "TOP", "Device" },
    ContainedBy { "Organization", "Organizational Unit" },
    NamedBy    { "CN" },
    MustContain { "ipHostNumber" },
    MayContain { "manager" },
    ASN1ObjID  {2 16 840 1 113719 1 167 6 4 1}
}

-- Abstraction of a network. The distinguished value of the cn attribute
-- denotes the network's canonical name
"ipNetwork" OBJECT-CLASS ::=
{
    Operation  ADD,
    Flags      { DS_EFFECTIVE_CLASS },
    SubClassOf { "TOP" },
    ContainedBy { "Organization", "Organizational Unit" },
    NamedBy    { "CN" },
    MustContain { "CN", "ipNetworkNumber" },
    MayContain { "Description", "L", "manager", "ipNetmaskNumber" },
    ASN1ObjID  {1 3 6 1 1 1 2 7}
}

-- Abstraction of a netgroup. May refer to other netgroups
"nisNetgroup" OBJECT-CLASS ::=
{
    Operation  ADD,
    Flags      { DS_EFFECTIVE_CLASS },
    SubClassOf { "TOP" },
    ContainedBy { "Organization", "Organizational Unit" },
    NamedBy    { "CN" },
    MustContain { "CN" },
    MayContain { "Description", "nisNetgroupTriple", "memberNisNetgroup" },
    ASN1ObjID  {1 3 6 1 1 1 2 8}
}

-- An entry in a NIS map
"nisObject" OBJECT-CLASS ::=
{
    Operation  ADD,
    Flags      { DS_EFFECTIVE_CLASS },
    SubClassOf { "TOP" },
    ContainedBy { "Organization", "Organizational Unit" },
    NamedBy    { "CN", "nisMapName" },

```

```

        MustContain { "CN", "nisMapEntry", "nisMapName" },
        MayContain { "Description" },
ASN1ObjID {1 3 6 1 1 1 2 10}
}

-- A device with a MAC address
-- According to RFC 2307 this is an Auxiliary class and device should be used
-- as Structural class. Currently we won't be able to do that so we are making
-- this as structural itself.
"ieee802Device" OBJECT-CLASS ::=
{
    Operation ADD,
    Flags { DS_EFFECTIVE_CLASS },
    SubClassOf { "TOP", "Device" },
    ContainedBy { "Organization", "Organizational Unit" },
    NamedBy { "CN" },
    MayContain { "macAddress" },
ASN1ObjID {2 16 840 1 113719 1 167 6 5 1}
}

-- A device with boot parameters
-- According to RFC 2307 this is an Auxiliary class and device should be used
-- as Structural class. Currently we won't be able to do that so we are making
-- this as structural itself.
"bootableDevice" OBJECT-CLASS ::=
{
    Operation ADD,
    Flags { DS_EFFECTIVE_CLASS },
    SubClassOf { "TOP", "Device" },
    ContainedBy { "Organization", "Organizational Unit" },
    NamedBy { "CN" },
    MayContain { "bootFile", "bootParameter" },
ASN1ObjID {2 16 840 1 113719 1 167 6 6 1}
}

END

```


Appendix B (*Oracle Internet Directory RFC2307 LDIF File*)

```
dn: cn=subschemasubentry
changetype: modify
add: attributetypes
attributetypes: ( 1.3.6.1.1.1.1.0
                  NAME 'uidNumber'
                  DESC 'An integer uniquely identifying a user in an administrative domain'
                  EQUALITY integerMatch
                  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
                  SINGLE-VALUE )
```

```
dn: cn=subschemasubentry
changetype: modify
add: attributetypes
attributetypes: ( 1.3.6.1.1.1.1.1
                  NAME 'gidNumber'
                  DESC 'An integer uniquely identifying a group in an administrative domain'
                  EQUALITY integerMatch
                  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
                  SINGLE-VALUE )
```

```
dn: cn=subschemasubentry
changetype: modify
add: attributetypes
attributetypes: ( 1.3.6.1.1.1.1.2
                  NAME 'gecos'
                  DESC 'The GECOS field; the common name'
                  EQUALITY caseIgnoreIA5Match
                  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
                  SINGLE-VALUE )
```

```
dn: cn=subschemasubentry
changetype: modify
add: attributetypes
attributetypes: ( 1.3.6.1.1.1.1.3
                  NAME 'homeDirectory'
                  DESC 'The absolute path to the home directory'
                  EQUALITY caseExactIA5Match
                  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
                  SINGLE-VALUE )
```

```
dn: cn=subschemasubentry
changetype: modify
add: attributetypes
attributetypes: ( 1.3.6.1.1.1.1.4
                  NAME 'loginShell'
                  DESC 'The path to the login shell'
                  EQUALITY caseExactIA5Match
                  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
```

SINGLE-VALUE)

dn: cn=subschemasubentry
changetype: modify
add: attributetypes
attributetypes: (1.3.6.1.1.1.1.5
 NAME 'shadowLastChange'
 EQUALITY integerMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE)

dn: cn=subschemasubentry
changetype: modify
add: attributetypes
attributetypes: (1.3.6.1.1.1.1.6
 NAME 'shadowMin'
 EQUALITY integerMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE)

dn: cn=subschemasubentry
changetype: modify
add: attributetypes
attributetypes: (1.3.6.1.1.1.1.7
 NAME 'shadowMax'
 EQUALITY integerMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE)

dn: cn=subschemasubentry
changetype: modify
add: attributetypes
attributetypes: (1.3.6.1.1.1.1.8
 NAME 'shadowWarning'
 EQUALITY integerMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE)

dn: cn=subschemasubentry
changetype: modify
add: attributetypes
attributetypes: (1.3.6.1.1.1.1.9
 NAME 'shadowInactive'
 EQUALITY integerMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE)

dn: cn=subschemasubentry
changetype: modify
add: attributetypes
attributetypes: (1.3.6.1.1.1.1.10
 NAME 'shadowExpire'
 EQUALITY integerMatch

SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
SINGLE-VALUE)

dn: cn=subschemasubentry
changetype: modify
add: attributetypes
attributetypes: (1.3.6.1.1.1.1.11
 NAME 'shadowFlag'
 EQUALITY integerMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE)

dn: cn=subschemasubentry
changetype: modify
add: attributetypes
attributetypes: (1.3.6.1.1.1.1.12
 NAME 'memberUid'
 DESC 'The uids of the users which belong to the group'
 EQUALITY caseExactIA5Match
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

dn: cn=subschemasubentry
changetype: modify
add: objectclasses
objectclasses: (1.3.6.1.1.1.2.0
 NAME 'posixAccount'
 SUP top
 AUXILIARY
 DESC 'Abstraction of an account with POSIX attributes'
 MUST (cn \$ uid \$ uidNumber \$ gidNumber \$ homeDirectory)
 MAY (userPassword \$ loginShell \$ gecos \$ description))

dn: cn=subschemasubentry
changetype: modify
add: objectclasses
objectclasses: (1.3.6.1.1.1.2.1
 NAME 'shadowAccount'
 SUP top
 AUXILIARY
 DESC 'Additional attributes for shadow passwords'
 MUST (uid)
 MAY (userPassword \$ shadowLastChange \$ shadowMin \$ shadowMax \$
shadowWarning \$ shadowInactive \$ shadowExpire \$ shadowFlag \$ description))

dn: cn=subschemasubentry
changetype: modify
add: objectclasses
objectclasses: (1.3.6.1.1.1.2.2
 NAME 'posixGroup'
 SUP top
 AUXILIARY
 DESC 'Abstraction of a group of accounts'
 MUST (gidNumber cn)

Deploying LDAP-UX with eDirectory, Oracle Internet Directory and OpenLDAP

MAY (userPassword \$ memberUid \$ description))

Appendix C (*DUAConfig Profile Definition for OpenLDAP*)

```

#
# This Schema file implements IETF Draft:
# "A Configuration Schema for LDAP Based Directory User Agents"
# draft-joslin-config-schema-06.txt
#
# URL: http://www.ietf.org/internet-drafts/draft-joslin-config-schema-06.txt
#
attributeType ( 1.3.6.1.4.1.11.1.3.1.1.3 NAME 'searchtimelimit' DESC 'Maximum time in
seconds a DUA should allow for a search to complete' EQUALITY integerMatch SYNTAX
1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE )

attributeType ( 1.3.6.1.4.1.11.1.3.1.1.11 NAME 'objectclassmap' DESC 'Objectclass mappings
used by a DUA' EQUALITY caseIgnoreIA5Match SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributeType ( 1.3.6.1.4.1.11.1.3.1.1.6 NAME 'authenticationmethod' DESC 'A keystring
which identifies the type of authentication method used to contact the DSA' EQUALITY
caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )

attributeType ( 1.3.6.1.4.1.11.1.3.1.1.13 NAME 'servicecredentiallevel' DESC 'Identifies type
of credentials a DUA should use when binding to the LDAP server for a specific service'
EQUALITY caseIgnoreIA5Match SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributeType ( 1.3.6.1.4.1.11.1.3.1.1.5 NAME 'followreferrals' DESC 'Tells DUA if it should
follow referrals returned by a DSA search result' EQUALITY caseIgnoreIA5Match SYNTAX
1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )

#attributeType ( 1.3.6.1.4.1.4203.1.3.5 NAME 'supportedFeatures' DESC 'features supported
by the server' EQUALITY objectIdentifierMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )

attributeType ( 1.3.6.1.4.1.11.1.3.1.1.0 NAME 'defaultserverlist' DESC 'Default LDAP server
host address used by a DUA' EQUALITY caseIgnoreMatch SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )

attributeType ( 1.3.6.1.4.1.11.1.3.1.1.10 NAME 'credentiallevel' DESC 'Identifies type of
credentials a DUA should use when binding to the LDAP server' EQUALITY
caseIgnoreIA5Match SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )

attributeType ( 1.3.6.1.4.1.11.1.3.1.1.15 NAME 'serviceauthenticationmethod' DESC
'Authentication method used by a service of the DUA' EQUALITY caseIgnoreMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

attributeType ( 1.3.6.1.4.1.11.1.3.1.1.7 NAME 'profilettl' DESC 'Time to live, in seconds,
before a client DUA should re-read this configuration profile' EQUALITY integerMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE )

attributeType ( 1.3.6.1.4.1.11.1.3.1.1.2 NAME 'preferredserverlist' DESC 'Preferred LDAP
server host addresses to be used by a DUA' EQUALITY caseIgnoreMatch SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )

```

```
attributeType ( 1.3.6.1.4.1.11.1.3.1.1.12 NAME 'defaultSearchScope' DESC 'Default search
scope used by a DUA' EQUALITY caseIgnoreIA5Match SYNTAX
1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
```

```
attributeType ( 1.3.6.1.4.1.11.1.3.1.1.1 NAME 'defaultsearchbase' DESC 'Default LDAP base
DN used by a DUA' EQUALITY distinguishedNameMatch SYNTAX
1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE )
```

```
attributeType ( 1.3.6.1.4.1.11.1.3.1.1.9 NAME 'attributemap' DESC 'Attribute mappings used
by a DUA' EQUALITY caseIgnoreIA5Match SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

```
attributeType ( 1.3.6.1.4.1.11.1.3.1.1.14 NAME 'servicessearchdescriptor' DESC 'LDAP search
descriptor list used by DUA' EQUALITY caseExactMatch SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 )
```

```
attributeType ( 1.3.6.1.4.1.11.1.3.1.1.4 NAME 'bindtimelimit' DESC 'Maximum time in
seconds a DUA should allow for the bind operation to complete' EQUALITY integerMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE )
```

```
objectClass ( 1.3.6.1.4.1.11.1.3.1.2.4 NAME 'DUAConfigprofile' DESC 'Abstraction of a base
configuration for a DUA' STRUCTURAL MUST cn MAY ( defaultserverlist $
preferredserverlist $ defaultsearchbase $ defaultSearchScope $ searchtimelimit
$ bindtimelimit $ credentiallevel $ authenticationmethod $ followreferrals $
servicessearchdescriptor $ servicecredentiallevel $ serviceauthenticationmethod $
objectclassmap $ attributemap $ profilettl ) )
```


Appendix D (*Sample DUAConfig Profile Entry*)

```

dn: cn=ldapuxprofile, ou=profiles,ou=ldap-ux,dc=acme,dc=com
objectClass: top
objectClass: duaconfigprofile
cn: ldapuxprofile
preferredserverlist: 192.1.1.1:389 192.1.1.2:444
defaultsearchbase: ou=ldap-ux,dc=acme,dc=com
searchtimelimit: 45
bindtimelimit: 5
authenticationmethod: simple
profilettl: 86400
credentiallevel: proxy anonymous
attributemap: passwd:userpassword=NULL*
attributemap: shadow:userpassword=NULL*
servicessearchdescriptor: passwd:ou=ldap-ux,dc=acme,dc=com?sub?(objectclass=posixaccount)
servicessearchdescriptor: shadow:
    ou=ldap-ux,dc=acme,dc=com?sub?(objectclass=shadowaccount)
servicessearchdescriptor: group:ou=ldap-ux,dc=acme,dc=com?sub?(objectclass=posixgroup)
servicessearchdescriptor: pam:ou=ldap-ux,dc=acme,dc=com?sub?(objectclass=posixaccount)
servicessearchdescriptor: rpc:ou=ldap-ux,dc=acme,dc=com?sub?(objectclass=oncrpc)
servicessearchdescriptor: protocols:ou=ldap-ux,dc=acme,dc=com?sub?(objectclass=ipprotocol)
servicessearchdescriptor: networks:ou=ldap-ux,dc=acme,dc=com?sub?(objectclass=ipnetwork)
servicessearchdescriptor: hosts:ou=ldap-ux,dc=acme,dc=com?sub?(objectclass=iphost)
servicessearchdescriptor: services:ou=ldap-ux,dc=acme,dc=com?sub?(objectclass=ipservice)
servicessearchdescriptor: netgroup:
    ou=ldap-ux,dc=acme,dc=com?sub?(objectclass=nisnetgroup)

```