

# Openldap and Solaris 10

James Hartley

01/08/08

## Abstract

Deploying Openldap in the Solaris environment in a manner that is secure and maintainable poses challenges that are not faced by Linux. Solaris 10 does not support Openldap as bundled software. Instead, Sun Microsystems ( and OpenSolaris ), provides it's own server and client architecture. In order to support Openldap, the Openldap software must be installed, and a choice of client software needs to be made. LDAP is not software. It is a protocol standard. As such, vendors and software developers are free to implement the standards in any manner in which they choose. This freedom comes with a price. Unlike simple naming services (like NIS), LDAP is a general directory service and it's implementation and configuration can be quite complex. Because of the liberal license allowed to implementers, many servers do not interact well with one another causing difficulties; if one wishes to use a directory services in creative ways, such as using Openldap instead of Sun's Native Directory Services to authenticate users. The freedom given to developers lead to other difficulties if one finds it necessary to replicate or synchronized databases between dis-similar server implementations. This work examines in detail how to implement a working example of Openldap on a Solaris 10 server using Solaris Native Client to provide LDAP services to Solaris clients and using Openldap Client on Linux. This work is driven by the scarcity of complete information found on the subject. In this paper we rely heavily on work completed on Solaris 9 by various contributors and give references to the work. It is hoped that with the information contained within, a novice can produce a working example system without any missing information. Armed with a working system it is much easier to learn the fundamentals and master the complexity of Openldap.

## Preparation:

In order to effectively incorporate Openldap into the Solaris environment it is important to configure the supporting services properly. These include the installation or configuration of DNS, Openldap server, Openldap client, and Certificate Management. How to maintain the server software and the support services is an important step that should be determined before deciding on the details of the implementation. Of course, a choice of server hardware needs to be determined and at least two servers will be required to support replication,

and synchronization. Each topic will be covered in turn with details on how to implement the configuration that provides a basic system.

In general our system demonstrates an Openldap configuration that supports tls:simple authentication, and provides login services to Solaris via Solaris native client. We use blast wave.org's implementation of openlap, because blastwave is compiled with Sun Microsystems Forte compiler suite, supports GCC options and is a natural choice for software development under Solaris. Blastwave's archive supports dependencies and does a good job of supporting shared libraries and library issues in general. Something that plagues those that attempt to compile openldap and all of the required support services. (footnote) If it is desired to build software from scratch see the companion document "Building OpenLDAP from Source".

Our topology of servers and clients will be the following hosts, their function and their operating system.

DNS domain name: bigsky.dom

elijah.bigsky.dom - ipaddr 192.168.1.21, master server and native ldap client, Solaris 10

jonah.bigsky.dom - ipaddr 192.168.1.20, slave server and native ldap client, Solaris 10

alphaomega.bigsky.dom - ipaddr 192.168.1.5, client, Linux (fedora core 6)

### **Preparing blastwave for Solaris clients/servers:**

The first step in successfully running openldap on Solaris 10 is to download and install a working openldap implementation. While there are several ways to do this, downloading from Blastwave provides many advantages. Blastwave is compiled with Forte, Sun's compiler suite. Blastwave archives track dependencies, so much like the Yum repositories on Linux, is is easy to upgrade and maintain the software. Blastwave maintainers are accessible via e-mail to answer questions about their builds. Hence, blastwave software is used to install and maintain the Openldap image. In order to work with Blastwave archives please read, then do, the steps described at the following URL, <http://www.blastwave.org/howto.html>.

### **Installing openldap from Blastwave:**

Once you have configured blastwave, then you can get openldap via the following command.

```
#pkg-get -i openldap
```

This will grab all the required, compatible support software for openldap as well as openldap itself. As noted in the howto, all software installs to the /opt/csw directory, so it is important to make sure that you have enough disk space on that partition to support user software. ( at least as much as you devote to

/usr/local for example, 2Gbytes is usually enough). Also noted in the howto, the standard Solaris package management routines are used by pkg-get to install the software. After installation you can verify that you have installed the software using the following command.

```
#pkginfo -l | grep CSW
```

all blastware packages have names that start with CSW as opposed to the standard Sun packages that have the prefix SUNW. Once openldap is installed in /opt/csw the rest of the configuration can begin. Be sure to install openldap on all systems running Solaris whether or not they are client or server systems. After installing the software make sure that both the root account and any user accounts that will use ldap client commands have /opt/csw/bin, and /opt/csw/sbin first in the PATH variable. This assures that the right openssl and openldap commands are being used.

If you plan to support either a Linux openldap client or Linux server in the mix verify that your Linux system has openldap installed. Most modern linux distributions come with openldap bundled into the system. We will not cover the installation of openldap on Linux system, but we will show how to interact with linux as a client to a Solaris openldap server.

### **Preparing the Master:**

In order to prepare the master it is important to realize that in Solaris we are attempting to configure two client systems on one host. Since we desire to authenticate users we need to decide which client we need to configure in order to support logins on the Solaris hosts. Solaris comes with it's own client. Solaris Native Client comes bundled with the Solaris 10 operating system. We have also installed openldap client on the system in the previous step. There are two ways to go; either configure the Solaris Native client to authenticate against the openldap server, or install nss\_ldap and pam\_ldap from PADL software (<http://www.padl.com>) and use the openldap client to authenticate against the server. The choice does affect the server in that we must support the native clients required schemas and Solaris specific system files or entries. We have made the choice of native client so we will now begin the process of preparing the master. Follow the steps below.

*Configure DNS* and /etc/hosts entries to include the ldap servers fully qualified names and aliases. ( see the appendix for the DNS configuration files if you wish to setup your own DNS server ). It is important to have the fully qualified name listed first in the /etc/hosts file according to [ref A]. Therefore the /etc/hosts entries look like

```
192.168.1.21 elijah.bigsky.dom elijah loghost
192.168.1.20 jonah.bigsky.dom jonah
```

on the elijah server, note that we did not remove the loghost alias from elijah in order for syslog to be unaffected.

*Configure the /etc/defaultdomain file* on all client systems running Solaris to have the name of the ldap domain that it is participating in. This is usually going to be the same domain as the DNS domain, and the base DN for the DIT tree. NSS uses this in conjunction with /etc/resolv.conf to use native client when authenticating. DNS domains are one of three standard topologies that are in common use in organizing DIT tree information in LDAP. So, in our example systems, the /etc/defaultdomain file contains the following entry.

```
bigsky.dom
```

the easiest method to populate this file is to run the following commands

```
#defaultdomain bigsky.dom
#defaultdomain > /etc/defaultdomain
```

*Configure the /etc/resolv.conf file* to search the bigsky.dom domain (optional). This line should look like;

```
search bigsky.dom
nameserver 192.168.1.21 *** on our systems the nameserver is the same as the
master. Change to your nameserver address.
```

*Set the library search path* using crle command - ( might be required for sudo and other things ). Use the following command

```
#crle -c /var/ld/ld.conf -l /lib:/usr/lib -s /lib/secure:/usr/lib/secure:/usr/lib/mps:/usr/lib/mps64
```

the entires for mps and mps/64 support “su” command using tls:simple (see <http://forum.java.sun.com/thread.jspa?threadID=5052764&messageID=9215595>)

*Create accounts, and set file permissions.* The default installation installs files and directory with root permissions. In order to run openldap with higher security an account other that root should be used. In this test environment the test account is “ldap” with an associated “ldap” group. Follow the steps below to create the accounts and change the ownership of any files and directories required by openldap.

Make the ldap directory and group accounts. The “ldap” directory will not have a login shell but it will be used to store various specific ldap scripts and files as they are built. The account can be used as storage for copies of configuration files and scripts but has no functional purpose other than providing a safe account for the execution of the openldap daemons.

```
#useradd -u55 -g55 -d /export/home/ldap -s /bin/false ldap
#groupadd -u55 -g55 ldap
#mkdir /export/home/ldap
#chown -R ldap:ldap /export/home/ldap
```

edit the `/etc/group` file and add “ldap” to the daemon group. After you are done the line for daemon should read  
daemon::12:root,ldap -note the “12” here is the number our system defined for the daemon group, your system might use a different number.

```
verify that the installation created the directory
/opt/csw/var/openldap-data – this is the default directory used by the back-
end database software if not make it with the following command
#mkdir -p /opt/csw/var/openldap-data, then
#chmod 700 /opt/csw/var/openldap-data - protect the directory for ldap only
access
#chown -R ldap:daemon /opt/csw/var/openldap-data - change ownership to
ldap,daemon
#chown -R ldap:daemon /opt/csw/etc/openldap - this is the default openldap
configuration directory, make sure ldap owns it.
#chmod 775 /opt/csw/var/run - this directory contains the pid of daemons that
are run, in particular it is used by openldap, so
#chown root:daemon /opt/csw/var/run - change ownership and perms so the
openldap daemon run as user ldap can write to the directory.
```

*Import required solaris schemas.* We require two schemas, `solaris.schema` and the `DUAConfigProfile.schema`. The `solaris.schema` supports Solaris specific attributes for native ldap version 1, and some NIS attributes. `DUAConfigProfile` supports native ldap version 2 attributes. These attributes do not contain the prefix “solaris” and are used by native ldap to authenticate to the LDAP server. We will be using a proxy profile within native ldap version 2 to authenticate via `tls:simple`. Therefore we need these schemas installed on our server. You can acquire and download these schemas at the urls:

```
http://web.singnet.com.sg/~garyttt/DUAConfigProfile.schema.txt and
http://web.singnet.com.sg/~garyttt/solaris.schema.txt
```

once these are downloaded, place them in the correct schema directory and change the permissions to `ldap:daemon`.

```
#cp DUAConfigProfile.schema.txt /opt/csw/etc/openldap/schema/DUACon-
figProfile.schema
#cp solaris.schema.txt /opt/csw/etc/openldap/schema/solaris.schema
#chown ldap:daemon /opt/csw/etc/openldap/DUAConfigProfile.schema
#chown ldap:daemon /opt/csw/etc/openldap/solaris.schema
```

```
#chmod 644 /opt/csw/etc/openldap/DUAConfigProfile.schema
#chmod 644 /opt/csw/etc/openldap/solaris.schema
```

*Build the Certificates for use by TLS to encrypt traffic* between the client and the server. This task requires the use of openssl. Openssl is a open source implementation of many common encryption algorithms in common use, plus some rather interesting uncommon algorithms. In this case we wish to construct a server certificate and a self signed certificate that can be used by the client to authenticate the server and begin encrypted traffic between the client and the server system. This is only one of many potential methods to secure communications between clients and servers. Consult [ref B], in order to get a much larger, and accurate picture of openldap security features. Once the certificates are built, openssl can be used to verify that the certificate can be passed to a client and read. Then TLS can be tested via the server before continuing any further server configuration. Since TLS is very finicky, certificate checking should be done before configuring the server to support the entire DIT tree. Once tested it is safe to configure and test a basic server setup and test the TLS function. Then, if successful, the server can be fully configured with the correct DIT information. Here are the steps.

Manually edit the file `/opt/csw/etc/ssl/openssl.cnf` - this file contains default values that will be used in the dn of the certificate. It is suggested that you manually edit this information to suit your organization and your base dn. In our case the organization is bigsky and the base dn is `dc=bigsky, dc=dom`. You can find a copy of the `openssl.cnf` file in the Appendix A. Note that the file has been edited to change the following parameters.

```
default_days = 3652
string_mask = MASK:0x2002
countryName_default = US
stateOrProvinceName_default = Nevada
localityName = locality Name (eg. city)
localityName_default = Vegas
0.organizationName_default = bigsky
```

explanations:

`default_days`, set high enough to get out of the way, reset in production

`string_mask`, set to support Netscape and plain text certificates.

we use `PrintableString+UTF8String` mask so if pure ASCII texts are used the resulting certificates are compatible with Netscape

`countryName_default`, self explanatory

`stateOrProvinceName_default`, self explanatory

`localityName_default`, self explanatory

`0.organizationName_default`, set to the first dc component of the base dn.

the base dn in our case is `dc=bigsky,dc=com`.

Beware that there is a soft link to this file in `/opt/csw/ssl`. By editing `/op-`

t/csw/etc/ssl/openssl.cnf you have changed the original file and hence the file pointed to by the soft-link.

*Create the self signed certificate* for the master server and the client. The best way to do this is to run the script “cr\_ssl\_certs\_openldap.sh” provided in the Appendix. This script is a modification of the script provided by [ref C]. Note the header information. Also note that this script is interactive and requires user input. Each step will be listed below with it’s required input. The script is placed in a working directory under the /export/home/ldap directory and run as root. It is note worthy to mention that the script creates a sub-directory called “demoCA” in the working directory and places the certificates in that directory. The script only creates the certificates, once created they must be manually moved into the appropriate openldap directory. Run the following command on the master server.

```
#!/cr_ssl_certs_openldap.sh
```

If the appropriate edits to the /etc/csw/etc/ssl/openssl.cnf file were made, this script will prompt with the default values you defined in that file. The only entries that you will have to input values for is the PEM pass phase, Common name, and email address. The script will strip the PEM pass phase from the certificate but you will have to input one when queried. Any pass phase will work in our example the pass phase was “secret”. The Common name needs to be the fully qualified name of the server. In this example that is “elijah@bigsky.dom”. Any value for the email address will do, but it should represent a responsible party. The script will prompt you twice for the default values, PEM pass phrase, Common name, and email address. Once this is complete the certificate and keys will be located in the sub-directory demoCA.

*Copy the certificates and key to the appropriate openldap directory.* The script will echo out some messages on where to copy files. Ignore these messages because the script is generic and assumes /etc/openldap is the ldap directory. For Blastwave the default directory is /opt/csw/etc/openldap, so the following command mv the certificates and keys to the proper locations.

```
#cd demoCA
#cp cacert.pem /opt/csw/etc/openldap/cacert.pem
#cp newcert.pem /opt/csw/etc/openldap/slapd-cert-ldap1.pem
#cp newreq.pem /opt/csw/etc/openldap/slapd-key-ldap1.pem
#cd /opt/csw/etc/openldap
#chown ldap:ldap *.pem
#chmod 640 slapd.-key-ldap1.pem
#chmod 644 cacert.pem
#chmod 644 slapd-cert-ldap1.pem
```

*Test the certificate with the openssl against the slapd server.* Since TLS encryption requires certificates and is the most finicky part of the configuration, test the TLS function and verify the certificate is valid before continuing. In order to do this, run slapd with the default slapd.conf file edited with the required lines to enable “Start\_TLS”. Then edit the openldap client file to use TLS when contacting the server. All testing is done on the master. Complete these steps to verify proper operation of both start-tls and ldaps.

```
#cd /opt/csw/etc/openldap
#cp slapd.conf.default slap.conf
edit slapd.conf and add the following lines to the global section of the file. ( the
part before the database declarations ).
TLSCipherSuite HIGH:MEDIUM:+TLSv1:+SSLv2:+SSLv3
TLSCACertificateFile /opt/csw/etc/openldap/cacert.pem
TLSCertificateFile /opt/csw/etc/openldap/slapd-cert-ldap1.pem
TLSCertificateKeyFile /opt/csw/etc/openldap/slapd-key-ldap1.pem
```

The server is now configured for “Start\_TLS” and ldaps. In order to test certificates start the server and attempt to connect to ldaps, then ldap. Follow these steps.

```
/opt/csw/libexec/slapd -u ldap -h “ldap:/// ldaps:///”
slapd should be running on both ports 389 and 636. Check to be sure the server
is in fact running. If the file permissions are set correctly the server should be
running. One common problem is that most implementations do not set the
permissions of the /opt/csw/var/run directory so that the “ldap” user can write
to the directory. Check permissions to make sure all of them are correct. Once
running issue the following command:
```

```
#openssl s_client -connect localhost:636 -showcerts
```

In the output you may ignore verify errors number 20, and 21 do a <ctrl>-c to exit. If you see the certificate you have successfully tested ldaps on port 636. Make sure that the fully qualified domain name shown by the CN entry is the same as the hostname in /etc/hosts file. When clients connect to the server they need to have the same fully qualified name defined in the client ldap configuration file.

*Test the openldap client against the openldap server using “Start\_TLS”.* This step is best done when there is some basic data in the DIT tree. Once data has been added into the DIT tree this step will be revisited.

*Kill the server for further configuration.*

```
#pkill slapd
```

*Edit the slapd.conf file to include the required schemas and directory information.* see the appendix for the slapd.conf file that is used to support user logins. Note that the two schemas, DUACfgProfile.schema, and solaris.scheme are included. Also note the back-end modules, the new ACL for proxyagent which we will explain later, and the database section entries for the dn, rootpw and database directory. Note the use of the database “bdb” which is only one of many potential back-ends supported by openldap. In this example the ACL’s are in the global section and the TLS values are in the database section. These should most likely be reversed in a production setup. ACLs tend to be specific to a database and TLS is most likely a global setup. In the case of one database it does not matter. Copy the slapd.conf file found in the appendix to the /opt/csw/etc/openldap/slapd.conf file Change the permission so that only the ldap user can read the file

```
#cp slapd.conf /opt/csw/etc/openlap/slapd.conf
#chown ldap:ldap
#chmod 600 /opt/csw/etc/openldap/slapd.conf
```

*Edit the DB\_CONFIG file.* this file controls the tuning parameter for the back-end database. In this case “bdb” the default for Blastwave’s implementation.

```
#cd /opt/csw/var/openldap-data
#cp DB_CONFIG.example DB_CONFIG
#chown ldap:ldap DB_CONFIG
#chmod 600 DB_CONFIG
edit the file and add the following lines
set _cachesize 0 2684354561
set _lg_regionmax 262144
set _lg_bsize 2097152
set _flags DB_LOG_AUTOREMOVE
```

At this point the slapd server and the database back-ends are configured with the correct information to accept the user data. The proper schemas have been loaded, the back-end database, manager dn and password identified and the database dn established. It is now possible to load the DIT tree with profiles and users.

*Create a run control script for openldap.* In order to support openldap at boot time and to control stopping and starting the server. See the Appendix for the run controls script. Install the script in /etc/init.d/openldap.server, and set the permissions appropriately.

```
once the file is copied to /etc/init.d/openldap.server then do
#chmod 744 /etc/init.d/openldap.server
```

*Add data into the DIT for profiles, and users.* Recall that the purpose of

this exercise is to authenticate users. In order to do that, standard account information must be added to the LDAP directory. Additionally, native ldap uses profiles to bind to the directory and then search for account information in order to login users. In order to support users and profiles a number of scripts have been written to import ascii file information in LDIF format into the directory. All of these scripts can be found in the appendix. Here is a listing of the required scripts and files that will be used to construct the DIT tree:

```
cr_example_com_ldif.sh - creates the ldif for the higher level portion of the
DIT tree
People.ldif - the ldif file that defines a couple of test user accounts
group.ldif - the ldif file that defines a couple of test groups
openldap_add.sh - used in conjunction with rebuild_example_com.sh to build
the DIT in LDAP.
rebuild_example_com.sh - driver script that builds the DIT tree in LDAP.
mgr.pwd - manager password file
```

It is a wise idea to download all of these scripts into a common working scripts directory. Note that the mgr.pwd file must reside in the working scripts directory to support the manager password required by the ldapadd command used in the scripts. The directory /export/home/ldap owned by ldap:ldap is used for that purpose in our setup. Feel free to use another.

Examine each of the scripts in the Appendix. You will note that each is specific to the domain and server name. These scripts will have to be modified to support other environments. Make the suitable edits to support your server and dn naming convention before running the scripts. Comments in the scripts should aid in editing the correct values for your site. The file 'mgr.pwd' is used to contain the manager passwd for the ldap clients. This file contains the passwd "secret" to match the one in the slapd.conf file. If the file does not exist in the scripts directory then create it.

```
#echo "secret" > mgr.pwd
#chown ldap:daemon mgr.pwd
#chmod 640 mgr.pwd
```

Once these files are suitably edited the DIT tree can be built by running the commands

```
#/etc/init.d/openldap.server start
#./rebuild_example_com.sh
```

Note that rebuild\_example\_com.sh repeated calls openldap\_add.sh. Openldap.sh calls ldapadd with a simple bind passing the password for the root dn in the clear. Which is secure when run on the master to build the initial DIT tree. Use secure communications via the clients and TLS to maintain the DIT tree from a client over the network. Once the script is run there will be a series of messages indicating success when adding entries into the directory.

## Preparing and Testing Openldap and Native Ldap Clients:

*Test the openldap client against the openldap server using TLS.* The openldap clients installed with openldap read the `/opt/csw/etc/openldap.conf` file for information on how to connect to the ldap server. In this case edit the file and instruct the client to connect to the server using TLS. Testing is done using the `ldapsearch` command with the `-ZZ` switch. `ldapsearch` uses the ldap protocols to connect to the server. Do the following.

```
#cd /opt/csw/etc/openldap
#cp ldap.conf.default ldap.conf
#chown ldap:daemon
#chmod 644 ldap.conf
edit the file and add the lines
HOST elijah.bigsky.dom
BASE dc=bigsky,dc=dom
TLS_CACERT=/opt/csw/openldap/cacert.pem
```

now test the openldap client against the directory server. Again, this is being done locally on the master server once this works other network clients can be configured. First we test connectivity by using simple authentication (`-x`), then we test `simple:tls (-ZZ)`, finally we test `ldaps`. Copy the `openldap_search.sh` script into the script directory. See the appendix for the code.

```
#which ldapsearch - should be /opt/csw/bin/ldapsearch
#./openldap_search.sh - (should return all the objectclasses for the DIT tree)
#ldapsearch -x -LLL (should return entries starting with the base dn)
#ldapsearch -x -LLL -ZZ (should only return entries if tls:simple worked)
#ldapsearch -x -LLL -H ldaps://elijah.bigsky.dom (should return the same entries)
```

At this point we have tested both `ldaps` and `tls:simple` for the openldap client running on the master server. Now we need to configure and test the native ldap client. There are two things that need to be done in order to test native client with TLS. First, we need to grab and store the certificate and key in a format suitable for native ldap. Second, we need to run the native ldap client version of `ldapsearch`. In order to get the certificate and key do the following steps from a normal user account.

Find and remove any `cert8.db` and `key3.db` files in the `.mozilla` directory under the home account.

Run mozilla locally.

```
$/usr/sfw/bin/mozilla &
```

In the browser url window type

```
https://elijah.bigsky.dom:636 - in the dialog popup save the cert permanently  
click ok when prompted
```

exit the browser.

The certificate and key are saved in the files cert8.db and key3.db are saved in the .mozilla directory. As an example, the system we use has the cert and the key in the directory

```
/export/home/testacct/.mozilla/default/ykwks4so.slt
```

your location may be different. You may have to experiment to find which directory mozilla is using. Once the certificate and key are obtained from the server, copy these files to the /var/ldap directory and change the permissions appropriately.

```
#cp /export/home/testacct/.mozilla/default/ykwks4so.slt /var/ldap
#cp /export/home/testacct/.mozilla/default/ykwks4so.slt /var/ldap
#chmod 644 /var/ldap/cert8.db
#chmod 644 /var/ldap/key3.db
```

now we are ready to test native ldap client and tls on the master server. Here are the steps.

```
/usr/bin/ldapsearch -h elijah.bigsky.dom -p -b "" -s base -Z -P /var/ldap/cert8.db
“(objectclass=*)”
```

you should get four lines of output,

```
version:1
dn:
objectClass: top
objectClass: OpenLDAProotDSE
```

Recapping to this point. We have a server with a DIT tree that supports user accounts and profiles required by native client. We have configured the openldap client and the native ldap client to use TLS and read the DIT tree. Now we need to configure the nss service on Linux and Solaris to use ldap to allow authentication. On the Linux side this required that the nss service file /etc/ldap.conf be configured to use the ldap server. On the Solaris side the same thing is done via “/var/ldap/ldap\_client\_file” and “/var/ldap/ldap\_client\_cred”. Edits must be made to /etc/nsswitch.conf on both the linux and the Solaris clients to use openldap for naming services required by the login process.

### **Preparing Native LDAP Client to Authenticate Users:**

We have already imported the cert8.db and key3.db files from the server and placed these in /var/ldap. Now there are two additional files that need to be created. These are

```
/var/ldap/ldap_client_file  
and  
/var/ldap/ldap_client_cred
```

The first file contains all of the parameters necessary to find the server and locate the appropriate entries to log in a user. The second file contains the password of the “proxyagent” profile and the dn to use to bind to the server. The proxyagent profile is used to bind to the server and locate the user entries. While the entries in the “ldap\_client\_file” are straight forward the password entry in the “ldap\_client\_cred” file needs some explanation. The password is stored in {NS1} format. The libldap.so uses an algorithm to encrypt the proxyagent password so that it is not stored in plaintext in the “ldap\_client\_cred” file. When the password is required for authentication it is decrypted by libldap.so and set to the server in plaintext ( ok since we use tls authentication to encrypt the traffic ). Once the server receives the password it hashes the password and compares it to the hashed password stored in the directory. If they hashes match the dn for the proxyagent is allowed to bind to the server. In our example files the passwords for proxyagent indeed do match. Although the passwords are stored in different formats and do not look the same. As a note, NS1 format is required for the proxyagent password stored in the “ldap\_client\_cred” file. Hence, we need a method of changing this password. You will find this procedure in the Appendix.

To configure the native ldap client do the following:

copy the ldap\_client\_cred file and the ldap\_client\_file from the Appendix to the /var/ldap directory.

```
#chmod 400 ldap_client_cred ldap_client_file
```

despite notes the the contrary it is safe to copy or edit these files by hand. In fact you should do so when using the openldap server. Do not run the the ldapinit that comes with native ldap to create these files.

edit the the /etc/nsswitch.conf file to support ldap you need the following lines.

```
passwd: files ldap  
group: files ldap  
shadow: files ldap  
hosts: files dns  
ipnodes: files
```

These are pretty self explanatory, except that in Solaris 10 you must have the appropriate ipnodes entry or the ldapclient initialization will hang. Note that another reason that you do not wish to run the ldapclient command from native client is that it the /etc/nsswitch entries for hosts and ipnodes to look at LDAP first instead of DNS. Stick to manual edits and use the files provided here as templates for your system configuration. You can now flush the nscd entries, start the server and start the client to test authentication against the server.

Note that the nscd can be a friend or foe. Once the system is working properly nscd stores naming information in a cache to speed up name look-ups. But it is has old or invalid information, new configuration information will not be seen until the entries timeout. It is best to turn off the nscd daemon while configuration and testing is being done. Once the entries are properly configured, and seen by the system you should turn the service back on. Follow These steps

```
#/etc/init.d/nscd stop - turn off name cache daemon
#/etc/init.d/ldap.client stop - turn off native ldap client ( should be off initially
)
#/etc/init.d/openldap.server stop - turn off the ldap server
#/etc/init.d/openldap.server start - turn ldap server on
#ps -ef | grep slapd - verify slapd is running
#/etc/init.d/ldap.client start - turn on client
#ps -ef | grep ldap - verify ldap_cachemgr (client) is running
#grep ldap /etc/nsswitch.conf - verify you have
passwd: files ldap
shadow: files ldap
group: files ldap
in the /etc/nsswitch.conf file
```

Test the name service with the following commands,

```
#/usr/lib/ldap/ldap_cachemgr -g
should yeild no errors, and identify elijah.bigsky.dom as the server, status should
be UP.
#id beowulf1
should yeild information about beowulf1 account
uid=206(beowulf1) gid=10(staff)
#getent passwd beowulf1 - yeilds
beowulf1:x:206:10:Beowulf1 account known the world over:/export/home/beowulf1:/bin/bash
#ldaplist -l passwd beowulf
```

```
dn: uid=beowulf1,ou=People,dc=bigsky,dc=dom
    givenName: beowulf1
    sn: beowulf1
    loginShell: /bin/bash
    uidNumber: 206
    gidNumber: 10
    objectClass: top
    objectClass: person
    objectClass: organizationalPerson
    objectClass: inetOrgPerson
    objectClass: posixAccount
    objectClass: shadowAccount
    uid: beowulf1
    cn: beowulf1
    homeDirectory: /export/home/beowulf1
    shadowLastChange: -1
    shadowMin: -1
    shadowMax: 99999
    shadowWarning: 7
    shadowInactive: -1
```

```
shadowExpire: -1
shadowFlag: 0
gecos: Beowulf1 account known the world over
userPassword: {CRYPT}IithjdruSkgos
```

If everything worked you should now be able to log into the beowulf1 account. The password for the beowulf1 account is “bobwab1@#” so if everything is working you should simply be able to “su” into the account from an normal user account.

```
$su - beowulf1
password: bobwab1@#
-bash 3.00$ id
uid=206(beowulf) gid=10(staff)
```

at this point everything works for a solaris native client.

### Preparing and Test Linux Clients (Fedora Core 6):

The file /etc/ldap.conf is used to configure nss to use openlap on linux systems. Hence, we have two ldap.conf files their purpose is different. I would have been nice if the name of the /etc/ldap.conf file had been more descriptive of it's purpose. This is the configuration file for the LDAP nameservice switch library and the LDAP PAM module. As such it defines parameters that are required to allow the PAM module to use the server to login users. Copy the /etc/ldap.conf file from the Appendix to the /etc/ldap.conf file on your linux system and copy the servers certificate to the /etc/openldap/cacerts directory. Then make the edits to the /etc/nsswitch.conf file to support ldap. After this you are ready to reboot and test.

copy by whatever means the certificate on the master server to the fedora client  
The require file is

```
/opt/csw/etc/openldap/cacert.pem copy to
/etc/openldap/cacerts/cacert.pem
this is the location defined in the /etc/ldap.conf file.
copy the /etc/ldap.conf file from the appendix to /etc/ldap.conf on the linux
client
make the following changes to the nsswitch.conf file
```

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

then reboot and test

```
#reboot....
#su - beowulf1
password: bobwab1@#
```

bash-3.00\$

done.

## Appendix A: /opt/csw/etc/ssl/openssl.cnf

```
#
# OpenSSL example configuration file.
# This is mostly being used for generation of certificate requests.
#
# NOTE: this file has been modified to reflect the values
# for our site... See Gary Tay's web page for details.
# This definition stops the following lines choking if HOME isn't
# defined.
HOME = .
RANDFILE = $ENV::HOME/.rnd

# Extra OBJECT IDENTIFIER info:
#oid_file = $ENV::HOME/.oid
oid_section = new_oids

# To use this configuration file with the "-extfile" option of the
# "openssl x509" utility, name here the section containing the
# X.509v3 extensions to use:
# extensions =
# (Alternatively, use a configuration file that has only
# X.509v3 extensions in its main [= default] section.)

[ new_oids ]

# We can add new OIDs in here for use by 'ca' and 'req'.
# Add a simple OID like this:
# testoid1=1.2.3.4
# Or use config file substitution like this:
# testoid2=${testoid1}.5.6

#####
[ ca ]
default_ca = CA_default # The default ca section

#####
[ CA_default ]

dir = ./demoCA # Where everything is kept
certs = $dir/certs # Where the issued certs are kept
crl_dir = $dir/crl # Where the issued crl are kept
database = $dir/index.txt # database index file.
#unique_subject = no # Set to 'no' to allow creation of
# several ctificates with same subject.
new_certs_dir = $dir/newcerts # default place for new certs.

certificate = $dir/cacert.pem # The CA certificate
serial = $dir/serial # The current serial number
crlnumber = $dir/crlnumber # the current crl number
# must be commented out to leave a V1 CRL
crl = $dir/crl.pem # The current CRL
private_key = $dir/private/cakey.pem # The private key
RANDFILE = $dir/private/.rand # private random number file

x509_extensions = usr_cert # The extensions to add to the cert

# Comment out the following two lines for the "traditional"
# (and highly broken) format.
name_opt = ca_default # Subject Name options
cert_opt = ca_default # Certificate field options

# Extension copying option: use with caution.
# copy_extensions = copy

# Extensions to add to a CRL. Note: Netscape communicator chokes on V2 CRLs
# so this is commented out by default to leave a V1 CRL.
# crlnumber must also be commented out to leave a V1 CRL.
# crl_extensions = crl_ext

default_days = 3652 # how long to certify for
default_crl_days = 30 # how long before next CRL
default_md = sha1 # which md to use.
preserve = no # keep passed DN ordering

# A few difference way of specifying how similar the request should look
# For type CA, the listed attributes must be the same, and the optional
# and supplied fields are just that :- )
policy = policy_match
```

```

# For the CA policy
[ policy_match ]
countryName           = match
stateOrProvinceName  = match
organizationName      = match
organizationalUnitName = optional
commonName            = supplied
emailAddress          = optional

# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.
[ policy_anything ]
countryName           = optional
stateOrProvinceName  = optional
localityName          = optional
organizationName      = optional
organizationalUnitName = optional
commonName            = supplied
emailAddress          = optional

#####
[ req ]
default_bits          = 1024
#added following line
default_md             = sha1
default_keyfile       = privkey.pem
distinguished_name    = req_distinguished_name
attributes            = req_attributes
x509_extensions       = v3_ca # The extensions to add to the self signed cert

# Passwords for private keys if not present they will be prompted for
# input_password = secret
# output_password = secret

# This sets a mask for permitted string types. There are several options.
# default: PrintableString, T61String, BMPString.
# pkix   : PrintableString, BMPString.
# utf8only: only UTF8Strings.
# nombstr : PrintableString, T61String (no BMPStrings or UTF8Strings).
# MASK:XXXX a literal mask value.
# WARNING: current versions of Netscape crash on BMPStrings or UTF8Strings
# so use this option with caution!
# we use PrintableString+UTF8String mask so if pure ASCII tests are used
# the resulting certificates are compatible with Netscape
string_mask = MASK:0x2002

# req_extensions = v3_req # The extensions to add to a certificate request

[ req_distinguished_name ]
countryName           = Country Name (2 letter code)
countryName_default   = US
countryName_min       = 2
countryName_max       = 2

stateOrProvinceName   = State or Province Name (full name)
stateOrProvinceName_default = Nevada

localityName           = Locality Name (eg. city)
localityName_default   = Vegas

0.organizationName    = Organization Name (eg, company)
0.organizationName_default = BigSky

# we can do this but it is not needed normally :- )
#1.organizationName   = Second Organization Name (eg, company)
#1.organizationName_default = World Wide Web Pty Ltd

organizationalUnitName = Organizational Unit Name (eg, section)
#organizationalUnitName_default =

commonName             = Common Name (eg, YOUR name)
commonName_max         = 64

emailAddress           = Email Address
emailAddress_max       = 64

# SET-ex3              = SET extension number 3

[ req_attributes ]
challengePassword      = A challenge password
challengePassword_min  = 4
challengePassword_max  = 20

unstructuredName       = An optional company name

[ usr_cert ]

# These extensions are added when 'ca' signs a request.

```

```

# This goes against PKIX guidelines but some CAs do it and some software
# requires this to avoid interpreting an end user certificate as a CA.

basicConstraints=CA:FALSE

# Here are some examples of the usage of nsCertType. If it is omitted
# the certificate can be used for anything *except* object signing.

# This is OK for an SSL server.
# nsCertType = server

# For an object signing certificate this would be used.
# nsCertType = objsign

# For normal client use this is typical
# nsCertType = client, email

# and for everything including object signing:
# nsCertType = client, email, objsign

# This is typical in keyUsage for a client certificate.
# keyUsage = nonRepudiation, digitalSignature, keyEncipherment

# This will be displayed in Netscape's comment listbox.
nsComment = "OpenSSL Generated Certificate"

# PKIX recommendations harmless if included in all certificates.
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer

# This stuff is for subjectAltName and issuerAltName.
# Import the email address.
# subjectAltName=email:copy
# An alternative to produce certificates that aren't
# deprecated according to PKIX.
# subjectAltName=email:move

# Copy subject details
# issuerAltName=issuer:copy

#nsCaRevocationUrl = http://www.domain.dom/ca-crl.pem
#nsBaseUrl
#nsRevocationUrl
#nsRenewalUrl
#nsCaPolicyUrl
#nsSslServerName

[ v3_req ]

# Extensions to add to a certificate request

basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment

[ v3_ca ]

# Extensions for a typical CA

# PKIX recommendation.
subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid:always,issuer:always

# This is what PKIX recommends but some broken software chokes on critical
# extensions.
#basicConstraints = critical,CA:true
# So we do this instead.
basicConstraints = CA:true

# Key usage: this is typical for a CA certificate. However since it will
# prevent it being used as an test self-signed certificate it is best
# left out by default.
# keyUsage = cRLSign, keyCertSign

# Some might want this also
# nsCertType = sslCA, emailCA

# Include email address in subject alt name: another PKIX recommendation
# subjectAltName=email:copy
# Copy issuer details
# issuerAltName=issuer:copy

# DER hex encoding of an extension: beware experts only!
# obj=DER:02:03
# Where 'obj' is a standard or added object
# You can even override a supported extension:
# basicConstraints= critical, DER:30:03:01:01:FF

```

```

[ crl_ext ]
# CRL extensions.
# Only issuerAltName and authorityKeyIdentifier make any sense in a CRL.
# issuerAltName=issuer:copy
# authorityKeyIdentifier=keyid:always,issuer:always

[ proxy_cert_ext ]
# These extensions should be added when creating a proxy certificate

# This goes against PKIX guidelines but some CAs do it and some software
# requires this to avoid interpreting an end user certificate as a CA.
basicConstraints=CA:FALSE

# Here are some examples of the usage of nsCertType. If it is omitted
# the certificate can be used for anything *except* object signing.

# This is OK for an SSL server.
# nsCertType = server

# For an object signing certificate this would be used.
# nsCertType = objsign

# For normal client use this is typical
# nsCertType = client, email

# and for everything including object signing:
# nsCertType = client, email, objsign

# This is typical in keyUsage for a client certificate.
# keyUsage = nonRepudiation, digitalSignature, keyEncipherment

# This will be displayed in Netscape's comment listbox.
# nsComment = "OpenSSL Generated Certificate"

# PKIX recommendations harmless if included in all certificates.
# subjectKeyIdentifier=hash
# authorityKeyIdentifier=keyid,issuer:always

# This stuff is for subjectAltName and issuerAltName.
# Import the email address.
# subjectAltName=email:copy
# An alternative to produce certificates that aren't
# deprecated according to PKIX.
# subjectAltName=email:move

# Copy subject details
# issuerAltName=issuer:copy

#nsCaRevocationUrl = http://www.domain.dom/ca-crl.pem
#nsBaseUrl
#nsRevocationUrl
#nsRenewalUrl
#nsCaPolicyUrl
#nsSslServerName

# This really needs to be in place for it to be a proxy certificate.
# proxyCertInfo=critical,language:id-ppl-anyLanguage,pathlen:3,policy:foo

```

## Appendix B: cr\_ssl\_certs\_opendap.sh

```

#!/bin/sh
#
# cr_ssl_certs_opendap.sh
#
# Purpose: - Create self-signed SSL Certs for OpenLDAP server
#
# Author: Gary Tay
#
# Date 6-Mar-2004
#
# Modified by: James Hartley
# Change date: 7/10/07
# Change: old version did not handle the current
# version of the default openssl.cnf found on
# solaris.
#
# Change date: 7/25/07
# added ${CATOP} variable to define the working directory
# commented out section on subjectAltName server aliases
# added sed command to include localityName_Default
#
# Change date: 12/09/07 - using blastwave server.
# as per gary's notes we have directly edited the
# openssl.cnf file appropriate for our system. Hence
# we comment out the edits and just create the certi-

```

```

# ficates for the server. See notes in the file
# /opt/csw/etc/ssl/openssl.cnf for this host.
#
# Change date: 12/12/07 - Added SSLCONF and SSLBIN
# environment variables to the script to point to
# our specific platform locations.
#
# Notes: run this version on solaris 10
# Notes: blastware software puts all installed soft-
# ware in /opt/csw... The original script only
# considers standard redhat and openldap installs
# into /usr/local. Please beware.
#
#####
#debug line
#set -x

#set following line to point to you systems openssl.cnf
#file
SSLCONFDIR=/opt/csw/etc/ssl
SSLBINDIR=/opt/csw/bin

#create the required directories for certificate creation
#this is a local directory. After the building the
#certificate move to the appropriate directory for your
#installation
CATOP=/demoCA
mkdir ${CATOP} >/dev/null 2>&1
mkdir ${CATOP}/certs >/dev/null 2>&1
mkdir ${CATOP}/crl >/dev/null 2>&1
mkdir ${CATOP}/newcerts >/dev/null 2>&1
mkdir ${CATOP}/private >/dev/null 2>&1
echo "01" > ${CATOP}/serial
cp /dev/null ${CATOP}/index.txt

# FIX IT
# we need to add better logic here to determine
# whether we are solaris or redhat, do after
# openldap works.

# Un-comment next two lines for RedHat
#cp /usr/share/ssl/openssl.cnf openssl.cnf
#ETC_OPENLDAP=/etc/openldap
# Un-comment next two lines for Others
#cp /usr/local/ssl/openssl.cnf ${CATOP}/openssl.cnf
#ETC_OPENLDAP=/usr/local/etc/openldap

# This is Gary's Original Code
#sed -e 's/GB/SG/' \
# -e 's/Berkshire/Singapore/' \
# -e 's/Newbury/Singapore/' \
# -e 's/My Company Ltd/Example Company Ltd/' \
# -e '/default_days/s/365/3652/' \
# ${CATOP}/openssl.cnf > ${CATOP}/openssl.cnf.new

# This is my New Code to handle Solaris defaults
# it is commented out because we have manually
# edited the openssl.cnf file for our system
#
#sed -e 's/AU/US/' \
# -e 's/Some-State/Nevada/' \
# -e '/Locality Name (eg, city)/a\
#localityName_default = Las Vegas' \
# -e 's/InTernet Widgits Pty Ltd/CACI/' \
# -e '/default_days/s/365/3652/' \
# ${CATOP}/openssl.cnf > ${CATOP}/openssl.cnf.new
#mv ${CATOP}/openssl.cnf.new ${CATOP}/openssl.cnf

#we do not have any aliases to worry about
#so leave commented out
#echo "" >>openssl.cnf
#echo "[ usr_cert ]" >> openssl.cnf
#echo "subjectAltName=DNS:ldap.'domainname',DNS:loadbalancer.'domainname'" >> openssl.cnf
#echo "" >> openssl.cnf

#####
#
# This is the actual working part of the script!
#
# Everything but ENV definitions, and creating
# the working directories, should have been commented
# out before these lines. This will be fixed
# after a working build of openldap is configured
#
#####

echo "Creating_CA_cert..."
echo "Please_enter_server's_FQDN_when_prompted_for_Common_Name:"
${SSLBINDIR}/openssl req -new -x509 -keyout ${CATOP}/private/akey.pem -out ${CATOP}/cacert.pem \
-days 3652 -config ${SSLCONFDIR}/openssl.cnf

```

```

echo "Creating_server_cert..."
echo "Please_enter_server'_s'_FQDN_when_prompted_for_Common_Name:"
${SSLBINDIR}/openssl req -new -x509 -nodes -keyout ${CATOP}/newreq.pem -out ${CATOP}/newreq.pem \
    -days 3652 -config ${SSLCONFDIR}/openssl.cnf

echo "Self_signing_server_cert..."
echo "Please_enter_server'_s'_FQDN_when_prompted_for_Common_Name:"
${SSLBINDIR}/openssl x509 -x509toreq -in ${CATOP}/newreq.pem -signkey ${CATOP}/newreq.pem -out ${CATOP}/tmp.pem

${SSLBINDIR}/openssl ca -config ${SSLCONFDIR}/openssl.cnf -policy policy_anything \
    -out ${CATOP}/newcert.pem -infiles ${CATOP}/tmp.pem
rm -f ${CATOP}/tmp.pem

#FIX IT -- when run as root have the option to install
#         for LDAP. Not implemented just yet... do a manual
#         move
#
echo "Please_copy_CA_Cert,_New_Cert_and_Key_to_OpenLDAP_config_dir..."
echo "using_the_following_commands"
echo "cp_demoCA/cacert.pem.$ETC_OPENLDAP"
echo "cp_demoCA/newcert.pem.$ETC_OPENLDAP/slapd-cert-ldap1.pem"
echo "cp_demoCA/newreq.pem.$ETC_OPENLDAP/slapd-key-ldap1.pem"
echo "chmod_640.$ETC_OPENLDAP/slapd-key-ldap1.pem"

# Uncomment for RedHat
#echo "chown ldap:ldap $ETC_OPENLDAP/*.pem"
# Uncomment for Others
#echo "chown ldap:daemon.$ETC_OPENLDAP/*.pem"
echo ""

```

## Appendix C: slapd.conf

```

#
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
include          /opt/csw/etc/openldap/schema/core.schema
include          /opt/csw/etc/openldap/schema/cosine.schema
include          /opt/csw/etc/openldap/schema/inetorgperson.schema
include          /opt/csw/etc/openldap/schema/nis.schema
# Solaris.schema provides nisDomainObject, absent from nis.schema
include          /opt/csw/etc/openldap/schema/solaris.schema
include          /opt/csw/etc/openldap/schema/DUAConfigProfile.schema
# Define global ACLs to disable default read access.

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

loglevel        4
pidfile         /opt/csw/var/run/slapd.pid
argsfile        /opt/csw/var/run/slapd.args

# Load dynamic backend modules:
modulepath      /opt/csw/libexec/openldap
moduleload      back_bdb.la
# moduleload    back_ldap.la
# moduleload    back_ldbm.la
# moduleload    back_passwd.la
# moduleload    back_shell.la

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

# Sample access control policy:
#
# Root DSE: allow anyone to read it
# Subschema (sub)entry DSE: allow anyone to read it
# Other DSEs:
#
#     Allow self write access
#     Allow authenticated users read access
#     Allow anonymous users to authenticate
# Directives needed to implement policy:
# access to dn.base="" by * read
# access to dn.base="cn=Subschema" by * read
# access to *
#     by self write
#     by users read
#     by anonymous auth
#
# if no access controls are present, the default policy
# allows anyone and everyone to read anything but restricts
# updates to rootdn. (e.g., "access to * by * read")
#

```

```

# rootdn can always read and write EVERYTHING!
##### different than gary's => allow bind_v2 bind_anon_dn
#
# My ACL directives
#
access to attrs=userPassword
    by self write
    by dn="cn=proxyagent,ou=profile,dc=bigsky,dc=dom" read
    by * auth
access to *
    by self write
    by * read

#####
# BDB database definitions
#####

database            bdb
suffix              "dc=bigsky,dc=dom"
rootdn              "cn=Manager,dc=bigsky,dc=dom"
# Cleartext passwords, especially for the rootdn, should
# be avoided. See slappasswd(8) and slapd.conf(5) for details.
# Use of strong authentication encouraged.
rootpw secret
# The database directory MUST exist prior to running slapd AND
# should only be accessible by the slapd and slap tools.
# Mode 700 recommended.
directory /opt/csw/var/openldap-data
# Indices to maintain
#Performance Tuning Directives
sizelimit 5000
threads 8
idletimeout 14400
cachesize 10000
checkpoint 256 15

TLS_CIPHER_SUITE HIGH:MEDIUM:+TLSv1:+SSLv2:+SSLv3
TLS_CACERTIFICATEFILE /opt/csw/etc/openldap/cacert.pem
TLS_CERTIFICATEFILE /opt/csw/etc/openldap/slapd-cert-ldap1.pem
TLS_CERTIFICATEKEYFILE /opt/csw/etc/openldap/slapd-key-ldap1.pem

#Use the following if client authentication is required
#TLSVerifyClient demand
# ... or not desired at all
#TLSVerifyClient never

```

## Appendix D: openldap.server

```

#!/bin/sh
#####
#
# openldap.server - Openldap start scripts
#
# uabashedly ripped off from Gary Tay see...
#
# http://web.singnet.com.sg/~garyttt for details
#
# Rewritten by James Hartley
#
# Date 06/14/06
#
# To do: --- rewrite to use the pid file to kill
#         the slapd daemon
# Modified -- 12/13/07
# for use in the domain bigsky.dom, using blastwave
# the blastwave directories are located under
# /opt/csw... not /usr/local...
#
#
#####
ETC_OPENLDAP_DIR=/opt/csw/etc/openldap
SLAPD_DIR=/opt/csw/libexec

# debug lines pick your own debug level
DEBUG=""
#DEBUG="-d 10"

case "$1" in
'start')
    if [ -f $ETC_OPENLDAP_DIR/slapd.conf -a -f $SLAPD_DIR/slapd ]; then
        echo "OpenLDAP slapd service starting."
        $SLAPD_DIR/slapd $DEBUG -u ldap -h "ldap:/// ldaps:///"
    fi
    ;;
'stop')

```

```

        PID=$(ps -ef | grep slapd | grep -v grep | awk '{print $2}')
        if [ -n $PID ]; then
            echo "OpenLDAP slapd service stopping"
            kill -INT $PID
        fi
        ;;
*)
    echo "Usage: $0 <start | stop >"
    exit 1
    ;;
esac

```

## Appendix E: rebuild\_example\_com.sh

```

#!/bin/sh
#####
#
# rebuild_example_com.sh -- Rebuilds the LDAP server
#
# ripped off from Gary Tey see --
#
# http://www.singnet.com.sg/~garyttt for details
#
# Script modified for my domain
#
# Author James Hartley
#
# Date 06/14/06
#
#####
set -x

OPENLDAP_DATA_DIR=/opt/csw/var/openldap-data

echo "WARNING: LDAP Data in $OPENLDAP_DATA_DIR will be gzipped!!!"
echo " and rebuilt from scratch, make sure you know what it means"
echo "Press [ctrl-c] to abort, enter [Yes] to continue..."
read a_key
[ "$a_key" != "Yes" ] && exit 1
/etc/init.d/openldap.server stop

if [ -d "$OPENLDAP_DATA_DIR" ]; then
    gzip -f $OPENLDAP_DATA_DIR/*.bdb
    gzip -f $OPENLDAP_DATA_DIR/*.db.*[0-9]
    gzip -f $OPENLDAP_DATA_DIR/*.log.*[0-9]
    gzip -f $OPENLDAP_DATA_DIR/lock
else
    mkdir -p $OPENLDAP_DATA_DIR
    chmod 750 $OPENLDAP_DATA_DIR
    chown ldap:daemon $OPENLDAP_DATA_DIR
fi

/etc/init.d/openldap.server start
sleep 5

./cr_example_com_ldif.sh
./openldap_add.sh
./cr_People_ldif.sh
./openldap_add.sh
./cr_group_ldif.sh
./openldap_add.sh

```

## Appendix F: cr\_example\_com\_ldif.sh

```

#!/bin/sh
#
#####
# cr_example_com_ldif.sh
#
# Purpose: -- Create initial ldif entries for dc=lsnext,dc=us
# and populate the file "example_com.ldif" for use by another
# script. (see Dependencies)
#
# Author: Gary Tay
# Date: Sans 2004
# Dependencies:
# This script is called by "rebuild_example_com.sh"
# It is NOT designed to be run separately although
# no harm will come from doing so. This script produces the
# the file "example_com.ldif"
#
# Outputs:

```

```

# creates example_com.ldif, ldap.ldif, openldap.ldif
#
# Modified by: James Hartley
#
# Added header and cleaned up syntax
# Changed all occurrences of dc=example,dc=com
# to dc=lsnext,dc=us, to match our domain
# Changed servernames to sunv420.lsnext.com
#
# Changed all occurrences of dc=lsnext,dc=us, to
# dc=bigsky,dc=dom; for our domain and changed
# the name of the server to elijah.bigsky.dom.
# in addition changed the capitalization to
# to match what work in the cr_example_com_ldif.sh
# file we had on palace (Bechtel's server)
#
#####
# OpenLDAP initial root entries

cat << EOF > example_com.ldif
dn: dc=bigsky,dc=dom
objectClass: top
objectClass: domain
objectClass: nisDomainObject
nisDomain: bigsky.dom
objectClass: dcObject
o: Bigsky
dc: bigsky

dn: cn=Manager,dc=bigsky,dc=dom
objectClass: organizationalRole
cn: Manager

dn: ou=People,dc=bigsky,dc=dom
objectClass: organizationalUnit
ou: People

dn: ou=group,dc=bigsky,dc=dom
objectClass: organizationalUnit
ou: group

dn: ou=profile,dc=bigsky,dc=dom
ou: profile
objectClass: top
objectClass: organizationalUnit

dn: cn=proxyagent,ou=profile,dc=bigsky,dc=dom
cn: proxyagent
sn: proxyagent
objectClass: top
objectClass: person
userPassword: {CRYPT}114aeXtphVSUg

dn: cn=sol8profile,ou=profile,dc=bigsky,dc=dom
objectClass: top
objectClass: SolarisNamingProfile
SolarisLDAPServers: 192.168.1.21
SolarisBindDN: cn=proxyagent,ou=profile,dc=bigsky,dc=dom
SolarisBindPassword: {NS}ecfa88f3a945c411
SolarisSearchBaseDN: dc=bigsky,dc=dom
SolarisAuthMethod: NS_LDAP_AUTH_NONE
SolarisTransportSecurity: NS_LDAP_SEC_NONE
SolarisSearchReferral: NS_LDAP_FOLLOWREF
SolarisSearchScope: NS_LDAP_SCOPE_ONELEVEL
SolarisSearchTimeLimit: 30
SolarisCacheTTL: 43200
cn: sol8profile

dn: cn=sol9profile,ou=profile,dc=bigsky,dc=dom
objectClass: DUAConfigProfile
defaultServerList: elijah.bigsky.dom
defaultSearchBase: dc=bigsky,dc=dom
authenticationMethod: simple
followReferrals: TRUE
defaultSearchScope: one
searchTimeLimit: 30
profileTTL: 43200
cn: sol9profile
credentialLevel: proxy
bindTimeLimit: 2

dn: cn=default,ou=profile,dc=bigsky,dc=dom
objectClass: DUAConfigProfile
defaultServerList: elijah.bigsky.dom
defaultSearchBase: dc=bigsky,dc=dom
authenticationMethod: simple
followReferrals: TRUE
defaultSearchScope: one
searchTimeLimit: 30
profileTTL: 43200
cn: default

```

```

credentialLevel: proxy
bindTimeLimit: 2

dn: cn=tls_profile,ou=profile,dc=bigsky,dc=dom
objectClass: top
objectClass: DUAConfigProfile
defaultServerList: elijah.bigsky.dom
defaultSearchBase: dc=bigsky,dc=dom
authenticationMethod: tls:simple
followReferrals: FALSE
defaultSearchScope: one
searchTimeLimit: 30
profileTTL: 43200
bindTimeLimit: 10
cn: tls_profile
credentialLevel: proxy
serviceSearchDescriptor: passwd: ou=People,dc=bigsky,dc=dom
serviceSearchDescriptor: group: ou=group,dc=bigsky,dc=dom
serviceSearchDescriptor: shadow: ou=People,dc=bigsky,dc=dom
serviceSearchDescriptor: netgroup: ou=netgroup,dc=bigsky,dc=dom

EOF
cp example_com.ldif ldap_add.ldif
cp example_com.ldif openldap_add.ldif

```

## Appendix G: People.ldif

```

dn: uid=beowulf1,ou=People,dc=bigsky,dc=dom
givenName: beowulf1
sn: beowulf1
loginShell: /bin/bash
uidNumber: 206
gidNumber: 10
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetorgperson
objectClass: posixAccount
objectClass: shadowAccount
uid: beowulf1
cn: beowulf1
homeDirectory: /export/home/beowulf1
shadowLastChange: -1
shadowMin: -1
shadowMax: 99999
shadowWarning: 7
shadowInactive: -1
shadowExpire: -1
shadowFlag: 0
gecos: Beowulf1 account known the world over
userPassword:{CRYPT}IithjdrusKgos

dn: uid=beowulf2,ou=People,dc=bigsky,dc=dom
givenName: beowulf2
sn: beowulf2
loginShell: /bin/bash
uidNumber: 207
gidNumber: 10
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetorgperson
objectClass: posixAccount
objectClass: shadowAccount
uid: beowulf2
cn: beowulf2
homeDirectory: /spare/beowulf2
shadowLastChange: -1
shadowMin: -1
shadowMax: 99999
shadowWarning: 7
shadowInactive: -1
shadowExpire: -1
shadowFlag: 0
gecos: Beowulf account known the world over
userPassword:{CRYPT}irq5Z4wp5XMnk

```

## Appendix H: group.ldif

```

dn: cn=staff,ou=group,dc=bigsky,dc=dom
cn: staff
gidNumber: 10
objectClass: top
objectClass: posixGroup

dn: cn=Users,ou=group,dc=bigsky,dc=dom

```

```
cn: Users
gidNumber: 102
objectClass: top
objectClass: posixGroup
```

## Appendix I: `openldap_add.sh`

```
#!/bin/sh
#####
#
# openldap_add.sh -- runs ldapadd on openldap_add.ldif
#
# again ripped off from Gray Tey see --
#
# http://www.singnet.com.sg/~garyttt for the details
#
# Modified by: James Hartley
# changed BINDIR path to reflect blastware's openldap
# basedir /opt/csw...
#
# changed the cn, dn's to reflect the new domain
# bigsky.dom
#
#####
BINDIR=/opt/csw/bin
if [ -f mgr.pwd ]; then
    $BINDIR/ldapadd -c -x -D "cn=Manager,dc=bigsky,dc=dom" -w 'cat mgr.pwd' -f openldap_add.ldif
else
    echo "mgr.pwd not found"
    echo "Please enter LDAP password for cn=Manager,dc=bogus,dc=net...."
    $BINDIR/ldapadd -c -x -D "cn=Manager,dc=bigsky,dc=dom" -W -f openldap_add.ldif
fi
```

## Appendix J: `openldap_search.sh`

```
#####
#
# openldap_search.sh -- list out objectclasses
#
# another shamelessly ripped off script from Gary Teh
#
# see http://www.singnet.com.sg/~garyttt
#
# Author James Hartley
#
# Date 6/15/06
#
#####
set -x
BINDIR=/opt/csw/bin
$BINDIR/ldapsearch -x -LLL -D"cn=Manager,dc=bigsky,dc=dom" -w 'cat mgr.pwd' "(objectclass=*)"
```

## Appendix K: `/var/ldap/ldap_client_cred`

```
#
# Do not edit this file manually; your changes will be lost. Please use ldapclient (1M) instead.
#
NS_LDAP_BINDDN= cn=proxyagent,ou=profile,dc=bigsky,dc=dom
NS_LDAP_BINDPASSWD= {NS1}ecfa88f3a945c411
```

## Appendix L: `/var/ldap/ldap_client_file`

```
#
# Do not edit this file manually; your changes will be lost. Please use ldapclient (1M) instead.
#
NS_LDAP_FILE_VERSION= 2.0
NS_LDAP_SERVERS= elijah.bigsky.dom
NS_LDAP_SEARCH_BASEDN= dc=bigsky,dc=dom
NS_LDAP_AUTH= tls:simple
NS_LDAP_SEARCH_REF= FALSE
NS_LDAP_SEARCH_SCOPE= one
NS_LDAP_SEARCH_TIME= 30
NS_LDAP_CACHETTTL= 43200
NS_LDAP_PROFILE= tls_profile
NS_LDAP_CREDENTIAL_LEVEL= proxy
NS_LDAP_SERVICE_SEARCH_DESC= passwd: ou=People,dc=bigsky,dc=dom
NS_LDAP_SERVICE_SEARCH_DESC= group: ou=group,dc=bigsky,dc=dom
```

```
NS_LDAP_SERVICE_SEARCH_DESC= shadow: ou=People,dc=bigsky,dc=dom
NS_LDAP_SERVICE_SEARCH_DESC= netgroup: ou=netgroup,dc=bigsky,dc=dom
NS_LDAP_BIND_TIME= 10
```

## Appendix M: /etc/ldap.conf

```
# @(#) $Id: ldap.conf,v 1.37 2006/04/13 03:26:17 lukeh Exp $
#
# This is the configuration file for the LDAP nameservice
# switch library and the LDAP PAM module.
#
# The man pages for this file are nss_ldap(5) and pam_ldap(5)
#
# PADL Software
# http://www.padl.com
#
# Your LDAP server. Must be resolvable without using LDAP.
# Multiple hosts may be specified, each separated by a
# space. How long nss_ldap takes to failover depends on
# whether your LDAP client library supports configurable
# network or connect timeouts (see bind_timelimit).
host elijah.bigsky.dom

# The distinguished name of the search base.
base dc=bigsky,dc=dom

# Another way to specify your LDAP server is to provide an
# uri with the server name. This allows to use
# Unix Domain Sockets to connect to a local LDAP Server.
#uri ldap://127.0.0.1/
#uri ldaps://127.0.0.1/
#uri ldapi://%2fvar%2frun%2fldapi_sock/
# Note: %2f encodes the '/' used as directory separator

# The LDAP version to use (defaults to 3
# if supported by client library)
#ldap_version 3

# The distinguished name to bind to the server with.
# Optional: default is to bind anonymously.
#binddn cn=proxyuser,dc=example,dc=com
binddn cn=proxyagent,ou=profile,dc=bigsky,dc=dom
bindpw password

# The credentials to bind with.
# Optional: default is no credential.
#bindpw secret

# The distinguished name to bind to the server with
# if the effective user ID is root. Password is
# stored in /etc/ldap.secret (mode 600)
#rootbinddn cn=manager,dc=example,dc=com
rootbinddn cn=Manager,dc=bigsky,dc=dom

# The port.
# Optional: default is 389.
#port 389

# The search scope.
#scope sub
#scope one
#scope base

# Search timelimit
#timelimit 30
timelimit 120

# Bind/connect timelimit
#bind_timelimit 30
bind_timelimit 120

# Reconnect policy: hard (default) will retry connecting to
# the software with exponential backoff, soft will fail
# immediately.
#bind_policy hard

# Idle timelimit; client will close connections
# (nss_ldap only) if the server has not been contacted
# for the number of seconds specified below.
#idle_timelimit 3600
idle_timelimit 3600

# Filter to AND with uid=%s
#pam_filter objectclass=account
pam_filter objectclass=posixAccount

# The user ID attribute (defaults to uid)
#pam_login_attribute uid
```

```

# Search the root DSE for the password policy (works
# with Netscape Directory Server)
#pam_lookup_policy yes

# Check the 'host' attribute for access control
# Default is no; if set to yes, and user has no
# value for the host attribute, and pam_ldap is
# configured for account management (authorization)
# then the user will not be allowed to login.
#pam_check_host_attr yes

# Check the 'authorizedService' attribute for access
# control
# Default is no; if set to yes, and the user has no
# value for the authorizedService attribute, and
# pam_ldap is configured for account management
# (authorization) then the user will not be allowed
# to login.
#pam_check_service_attr yes

# Group to enforce membership of
#pam_groupdn cn=PAM,ou=Groups,dc=example,dc=com

# Group member attribute
#pam_member_attribute uniquemember
#pam_member_attribute memberUid

# Specify a minimum or maximum UID number allowed
#pam_min_uid 0
#pam_max_uid 0

# Template login attribute, default template user
# (can be overridden by value of former attribute
# in user's entry)
#pam_login_attribute userPrincipalName
#pam_template_login_attribute uid
#pam_template_login nobody

# HEADS UP: the pam_crypt, pam_nds_passwd,
# and pam_ad_passwd_options are no
# longer supported.
#
# If you are using XAD, you can set pam_password
# to racf, ad, or exop. Make sure that you have
# SSL enabled.

# Do not hash the password at all; presume
# the directory server will do it, if
# necessary. This is the default.
#pam_password clear

# Hash password locally; required for University of
# Michigan LDAP server, and works with Netscape
# Directory Server if you're using the UNIX-Crypt
# hash mechanism and not using the NT Synchronization
# service.
#pam_password crypt
#pam_password crypt

# Remove old password first, then update in
# cleartext. Necessary for use with Novell
# Directory Services (NDS)
#pam_password clear_remove_old
#pam_password nds

# RACF is an alias for the above. For use with
# IBM RACF
#pam_password racf

# Update Active Directory password, by
# creating Unicode password and updating
# unicodePwd attribute.
#pam_password ad

# Use the OpenLDAP password change
# extended operation to update the password.
#pam_password exop

# Redirect users to a URL or somesuch on password
# changes.
#pam_password_prohibit_message Please visit http://internal to change your password.

# RFC2307bis naming contexts
# Syntax:
# nss_base_XXX          base?scope?filter
# where scope is {base,one,sub}
# and filter is a filter to be &'d with the
# default filter.
# You can omit the suffix eg:
# nss_base_passwd       ou=People,

```

```

# to append the default base DN but this
# may incur a small performance impact.
nss_base_passwd ou=People,dc=bigsky,dc=dom?one
nss_base_shadow ou=People,dc=bigsky,dc=dom?one
nss_base_group ou=Group,dc=bigsky,dc=dom?one
#nss_base_hosts ou=Hosts,dc=example,dc=com?one
#nss_base_services ou=Services,dc=example,dc=com?one
#nss_base_networks ou=Networks,dc=example,dc=com?one
#nss_base_protocols ou=Protocols,dc=example,dc=com?one
#nss_base_rpc ou=Rpc,dc=example,dc=com?one
#nss_base_ethers ou=Ethers,dc=example,dc=com?one
#nss_base_netmasks ou=Networks,dc=example,dc=com?one
#nss_base_bootparams ou=Ethers,dc=example,dc=com?one
#nss_base_aliases ou=Aliases,dc=example,dc=com?one
#nss_base_netgroup ou=Netgroup,dc=example,dc=com?one

# Just assume that there are no supplemental groups for these named users
nss_initgroups_ignoreusers root,ldap,named,avahi,haldaemon

# attribute/objectclass mapping
# Syntax:
#nss_map_attribute rfc2307attribute mapped_attribute
#nss_map_objectclass rfc2307objectclass mapped_objectclass

# configure --enable-nds is no longer supported.
# NDS mappings
#nss_map_attribute uniqueMember member

# Services for UNIX 3.5 mappings
#nss_map_objectclass posixAccount User
#nss_map_objectclass shadowAccount User
#nss_map_attribute uid msSFU30Name
#nss_map_attribute uniqueMember msSFU30PosixMember
#nss_map_attribute userPassword msSFU30Password
#nss_map_attribute homeDirectory msSFU30HomeDirectory
#nss_map_attribute homeDirectory msSFUHomeDirectory
#nss_map_objectclass posixGroup Group
#pam_login attribute msSFU30Name
#pam_filter objectclass=User
#pam_password ad

# configure --enable-mssfu-schema is no longer supported.
# Services for UNIX 2.0 mappings
#nss_map_objectclass posixAccount User
#nss_map_objectclass shadowAccount user
#nss_map_attribute uid msSFUName
#nss_map_attribute uniqueMember posixMember
#nss_map_attribute userPassword msSFUPassword
#nss_map_attribute homeDirectory msSFUHomeDirectory
#nss_map_attribute shadowLastChange pwdLastSet
#nss_map_objectclass posixGroup Group
#nss_map_attribute cn msSFUName
#pam_login attribute msSFUName
#pam_filter objectclass=User
#pam_password ad

# RFC 2307 (AD) mappings
#nss_map_objectclass posixAccount user
#nss_map_objectclass shadowAccount user
#nss_map_attribute uid sAMAccountName
#nss_map_attribute homeDirectory unixHomeDirectory
#nss_map_attribute shadowLastChange pwdLastSet
#nss_map_objectclass posixGroup group
#nss_map_attribute uniqueMember member
#pam_login attribute sAMAccountName
#pam_filter objectclass=User
#pam_password ad

# configure --enable-authpassword is no longer supported
# AuthPassword mappings
#nss_map_attribute userPassword authPassword

# AIX SecureWay mappings
#nss_map_objectclass posixAccount aixAccount
#nss_base_passwd ou=aixaccount,?one
#nss_map_attribute uid userName
#nss_map_attribute gidNumber gid
#nss_map_attribute uidNumber uid
#nss_map_attribute userPassword passwordChar
#nss_map_objectclass posixGroup aixAccessGroup
#nss_base_group ou=aixgroup,?one
#nss_map_attribute cn groupName
#nss_map_attribute uniqueMember member
#pam_login attribute userName
#pam_filter objectclass=aixAccount
#pam_password clear

# Netscape SDK LDAPS
#ssl on

# Netscape SDK SSL options

```

```

#sslpath /etc/ssl/certs
# OpenLDAP SSL mechanism
# start_tls mechanism uses the normal LDAP port, LDAPS typically 636
ssl_start_tls
#ssl on

# OpenLDAP SSL options
# Require and verify server certificate (yes/no)
# Default is to use libldap's default behavior, which can be configured in
# /etc/openldap/ldap.conf using the TLS_REQCERT setting. The default for
# OpenLDAP 2.0 and earlier is "no", for 2.1 and later is "yes".
#tls_checkpeer yes

# CA certificates for server certificate verification
# At least one of these are required if tls_checkpeer is "yes"
#tls_cacertfile /etc/ssl/ca.cert
#tls_cacertfile /etc/openldap/cacerts/cacert.pem
#tls_cacertdir /etc/ssl/certs

# Seed the PRNG if /dev/urandom is not provided
#tls_randfile /var/run/egd-pool

# SSL cipher suite
# See man ciphers for syntax
#tls_ciphers TLSv1

# Client certificate and key
# Use these, if your server requires client authentication.
#tls_cert
#tls_key

# Disable SASL security layers. This is needed for AD.
#sasl_secprops maxssf=0

# Override the default Kerberos ticket cache location.
#krb5_ccname FILE:/etc/.ldapcache

# SASL mechanism for PAM authentication - use is experimental
# at present and does not support password policy control
#pam_sasl_mech DIGEST-MD5

```