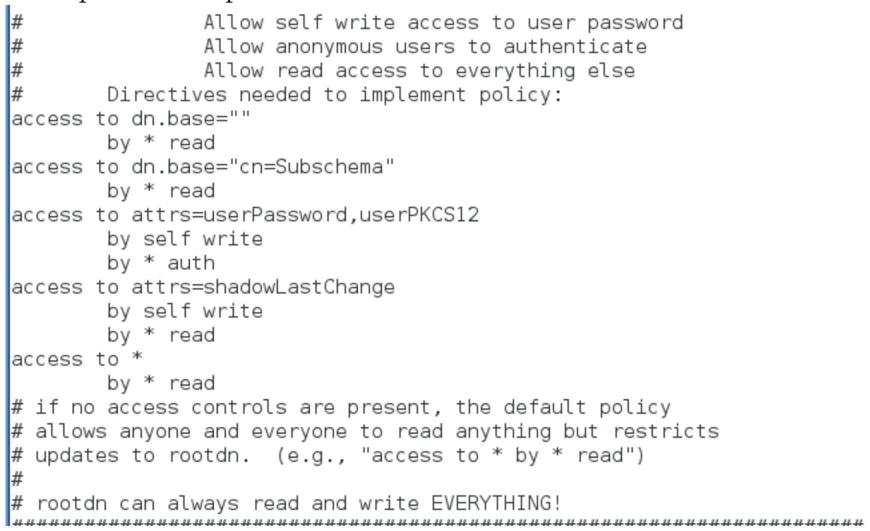
1 OpenLDAP Configuration

Once the OpenLDAP software has been built and installed, we are ready to configure it. The configuration is primarily accomplished through the slapd.conf file, normally found in the /usr/local/etc/openldap or /etc/openldap directory.

- The slapd.conf file consists of three types of configuration information: global, backend specific and database specific. Global information is specified first, followed by information associated with a particular backend type, which is then followed by information associated with a particular database instance. Global directives can be overridden in backend and/or database directives, and backend directives can be overridden by database directives.
- Blank lines and comment lines beginning with a # character are ignored.
- *Note:* If a line begins with white space, it is considered a continuation of the previous line (even if the previous line is a comment).
- Please take care of extra spaces and newlines. Extra white spaces or new line may lend you in trouble.

  1. Global directives:

     1.1 access to <what> [by <who> <accesslevel> <control>]+
     This directive grants access (specified by <accesslevel>) to a set of entries and/or attributes (specified by <what>) by one or more requesters (specified by <who>).
     A snapshot of slapd.conf:

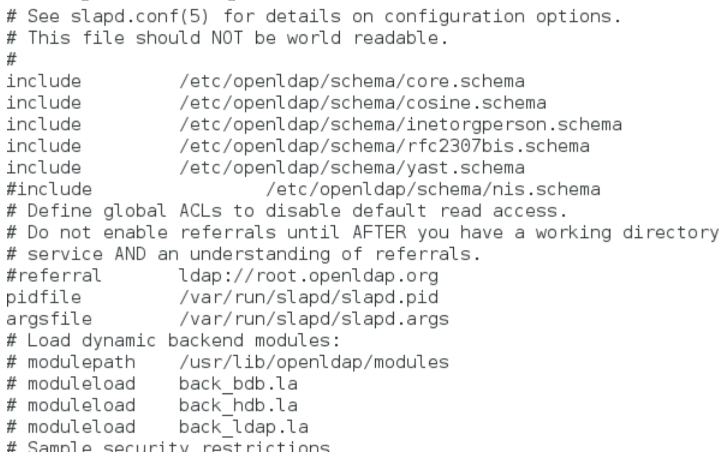


```
#               Allow self write access to user password
#               Allow anonymous users to authenticate
#               Allow read access to everything else
#       Directives needed to implement policy:
access to dn.base=""
        by * read
access to dn.base="cn=Subschema"
        by * read
access to attrs=userPassword,userPKCS12
        by self write
        by * auth
access to attrs=shadowLastChange
        by self write
        by * read
access to *
        by * read
# 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!
#######################################################################
```

1.2 attributetype <Attribute Type Description>
This directive defines an attribute type. We can specify new attributes also in schema file.

1.3 idletimeout <integer>
Specify the number of seconds to wait before forcibly closing an idle client connection. An idletimeout of 0, the default, disables this feature.

1.4 include <filename>
This directive specifies that slapd should read additional configuration information from the given file before continuing with the next line of the current file.
A snapshot of slapd.conf:



```
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/rfc2307bis.schema
include         /etc/openldap/schema/yast.schema
#include                /etc/openldap/schema/nis.schema
# Define global ACLs to disable default read access.
# Do not enable referrals until AFTER you have a working directory
# service AND an understanding of referrals.
#referral       ldap://root.openldap.org
pidfile         /var/run/slapd/slapd.pid
argsfile        /var/run/slapd/slapd.args
# Load dynamic backend modules:
# modulepath    /usr/lib/openldap/modules
# moduleload    back_bdb.la
# moduleload    back_hdb.la
# moduleload    back_ldap.la
# Sample security restrictions
```

1.5 loglevel <integer>
This directive specifies the level at which debugging statements and operation statistics should be syslogged.

2. Backend Directives:

2.1 backend <type>
This directive marks the beginning of a backend declaration. <type> should be one of the supported backend types.

3. Database Directives:

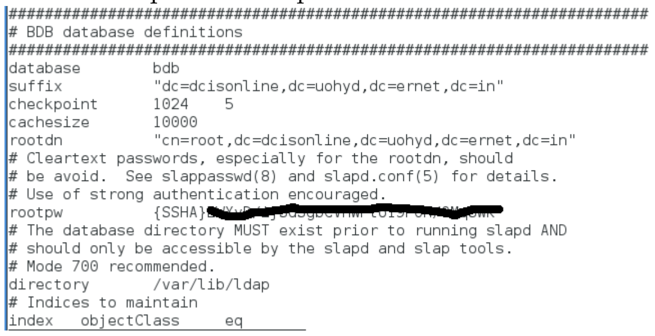
3.1 database <type>
This directive marks the beginning of a database instance declaration. <type> should be one of the supported backend types.

3.2 rootdn <DN>
This directive specifies the DN that is not subject to access control or administrative limit restrictions for operations on

this database. The DN need not refer to an entry in this database or even in the directory.

3.3 rootpw <password>
This directive can be used to specifies a password for the DN for the rootdn.
Note: Use slappasswd to generate password.

3.4 suffix <dn suffix>
This directive specifies the DN suffix of queries that will be passed to this backend database. Multiple suffix lines can be given, and at least one is required for each database definition. A snapshot of slapd.conf:



```
#######################################################################
# BDB database definitions
#######################################################################
database        bdb
suffix          "dc=dcisonline,dc=uohyd,dc=ernet,dc=in"
checkpoint      1024    5
cachesize       10000
rootdn          "cn=root,dc=dcisonline,dc=uohyd,dc=ernet,dc=in"
# Cleartext passwords, especially for the rootdn, should
# be avoid.  See slappasswd(8) and slapd.conf(5) for details.
# Use of strong authentication encouraged.
rootpw          {SSHA}[redacted]
# The database directory MUST exist prior to running slapd AND
# should only be accessible by the slapd and slap tools.
# Mode 700 recommended.
directory       /var/lib/ldap
# Indices to maintain
index   objectClass     eq
```

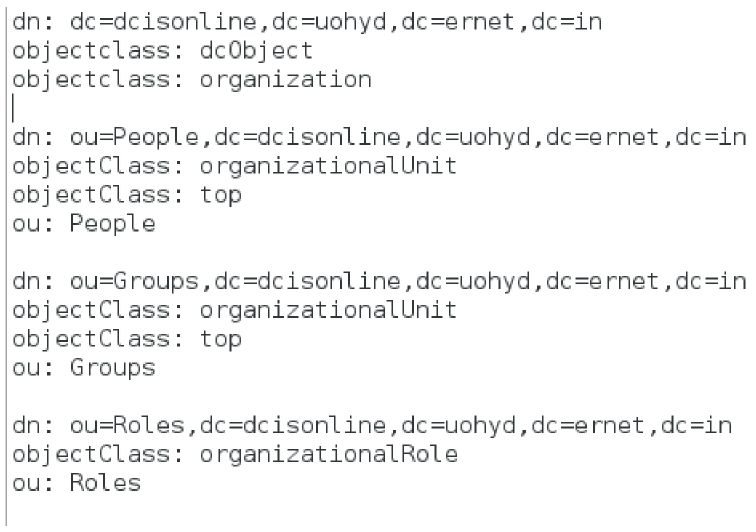
That's it! We are done with the LDAP configuration. for more info see the *man* pages.

2 LDAP user account creation

1. Initializing the base system

   To initialize the system we add initial.ldif file with ldapadd command, the input file format is LDIF(LDAP Data Interchange Format), which is of this form:
   # comment
   dn: <distinguished name>
   <attrdesc>: <attrvalue>
   <attrdesc>: <attrvalue>
   ...

   1.1 First of all we add initial.ldif to ldap server, an example snapshot of initial ldif is given below



```
dn: dc=dcisonline,dc=uohyd,dc=ernet,dc=in
objectclass: dcObject
objectclass: organization

dn: ou=People,dc=dcisonline,dc=uohyd,dc=ernet,dc=in
objectClass: organizationalUnit
objectClass: top
ou: People

dn: ou=Groups,dc=dcisonline,dc=uohyd,dc=ernet,dc=in
objectClass: organizationalUnit
objectClass: top
ou: Groups

dn: ou=Roles,dc=dcisonline,dc=uohyd,dc=ernet,dc=in
objectClass: organizationalRole
ou: Roles
```

Then run ldapadd to import the file.
ldapadd -x -D "cn=root,dc=dcisonline,dc=uohyd,dc=ernet,dc=in"
-f initial.ldif

1.2 Adding users to the system

Create users.ldif and add it to ldap server with ldapadd
ldapadd -x -D "cn=root,dc=dcisonline,dc=uohyd,dc=ernet,dc=in"
-f users.ldif

1.3 Migrating passwd or nis users

To do this we need some scripts, we use the migration perl scripts to import all users to ldif format and then add allusers.ldif to ldap server with ldapadd.
perl migrate_ passwd.pl -i /etc/passwd > allusers.ldif
and then
ldapadd -x -D "cn=root,dc=dcisonline,dc=uohyd,dc=ernet,dc=in"
-f allusers.ldif

3 Using LDAP server for authentication

Prerequisite:
We need libnss_ldap and libpam_ldap installed on clients to use ldap for authentication.

3.1 Configuring Clients to use ldap for authentication

If we want to use LDAP as central authentication server like nis server, We must configure OpenLDAP on each of the client systems, This also includes the server as it will likely be a client unto itself (i.e. it will access the LDAP server via localhost to obtain authentication information). To do this, you must edit the /etc/ldap.conf file. The entries we are most interested in are the following:

```
host 172.16.88.49 <— The address of ldap server
base dc=dcisonline,dc=uohyd,dc=ernet,dc=in
rootbinddn cn=root,dc=dcisonline,dc=uohyd,dc=ernet,dc=in
scope one
pam_filter objectclass=posixaccount
pam_login_attribute uid
pam_member_attribute gid
pam_password crypt <—in palce of crypt you can use md5,ssha or whatever
encryption your system is using, make sure no incompatible form,
otherwise it may lend you in trouble!
nss_base_passwd ou=People,dc=dcisonline,dc=uohyd,dc=ernet,dc=in?one
nss_base_shadow ou=People,dc=dcisonline,dc=uohyd,dc=ernet,dc=in?one
nss_base_group ou=Group,dc=dcisonline,dc=uohyd,dc=ernet,dc=in?one
```

Then edit /etc/nsswitch.conf

```
passwd: files ldap or compat ldap
group: files ldap
shadow: files ldap
```

3.2 GUI way of client configuration

Install auth-config or ldap-auth-config gui (on most of the linux versions) and set the host name and base as above.
For OpenSuse users:

1. Go to yast2
2. under Network services choose LDAP client
3. On new window (LDAP client cofiguration) choose radio button use ldap
4. Enter address of LDAP server and base DN
5. click OK.

That's it!

4 SSL certificate and TLS 1.0

SSL, or Secure Socket Layer, is a technology which allows web browsers and web servers to communicate over a secured connection. This means that the

data being sent is encrypted by one side, transmitted, then decrypted by the other side before processing. This is a two-way process, meaning that both the server AND the browser encrypt all traffic before sending out data.
There are two ways of enabling TLS in tomcat, 1- with openssl and 2- with keytool.
the tomcat server supports only JKS, PKCS11 or PKCS12 format keystores. The JKS format is Java's standard "Java KeyStore" format, and is the format created by the keytool command-line utility. This tool is included in the JDK. The PKCS12 format is an internet standard, and can be manipulated via OpenSSL.

- using keytool:

  To create a new keystore from scratch, containing a single self-signed Certificate we use following command:
  $ JAVA_HOME/bin/keytool –genkey –alias tomcat –keyalg RSA
  and specify a password value of 'changeit'.(The tomcat uses the default password as *changeit*, we can specify our own also but that we need to specify in configuration of tomcat)
  This command will create a new hidden file, as /userhomedir/.keystore.

- using OpenSSL:

  1. Create root ca
     To create a rootCA i have written a script createRootCA.sh, simply run the script, enter the self explanatory answers to questions and specify your private key(password) for RootCA, your root CA will be created.
  2. Create a CSR(Certificate Signing Request)
     openssl req –config openssl.cnf –new –nodes –keyout private/server.key –out server.csr –days 365
     Enter the answers to the self explanatory questions and your key for server certificate, on completion, two files - server.csr and private/server.key are created.
  3. Signing the csr:
     openssl ca –config openssl.cnf -policy policy_anything –out certs/server.crt –infiles server.csr
     Enter your root CA private when prompted.
  4. Import certificate to PKCS12:
     openssl pkcs12 –export –in server.crt –inkey server.key –out mycert.p12 –name tomcat –CAfile rootCA.crt –caname root –chain

5 Tomcat server configuration

The two most important configuration files to get Tomcat up and running are called server.xml and web.xml. By default, these files are located at TOMCAT-HOME/conf/server.xml and TOMCAT-HOME/conf/web.xml, respectively.

- Web.xml
  The web.xml file is derived from the servlet specification, and contains information used to deploy and configure the components of the web application. It is used to define default values for all contexts. If this method is utilized, Tomcat will use TOMCAT-HOME/conf/web.xml as a base configuration, which can be overwritten by application-specific WEB-INF/web.xml files.

  - In this file we can specify default welcome pages (e.g. index.jsp) to whatever we want like welcome.jsp or any default page.
  - we can change default time for session expire in web.xml default time is 30 min, we can have our own time set like 10 mins.
    ```
    <session-config>
    <session-timeout>10</session-timeout>
    </session-config>
    ```

- server.xml
  The server.xml file is Tomcat's main configuration file, and is responsible for specifying Tomcat's initial configuration on startup as well as defining the way and order in which Tomcat boots and builds. The elements of the server.xml file belong to five basic categories - Top Level Elements, Connectors, Containers, Nested Components, and Global Settings. All of the elements within these categories have many attributes that can be used to fine-tune their functionality. Most often, if you need to make any major changes to your Tomcat installation, such as specifying application port numbers, server.xml is the file to edit.

5.1 Enabling HTTPS

Tomcat can use two different implementations of SSL:

1. The JSSE implementation provided as part of the JRE

- Define the connector protocol in server.xml

```
<-- Define a blocking Java SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector protocol="org.apache.coyote.http11.Http11Protocol"
           port="8443" .../>

<-- Define a non-blocking Java SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector protocol="org.apache.coyote.http11.Http11NioProtocol"
           port="8443" .../>
```

- Uncomment the connector in server.xml

```
<-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<!--
<Connector
           port="8443" maxThreads="200"
           scheme="https" secure="true" SSLEnabled="true"
           keystoreFile="${user.home}/.keystore" keystorePass="changeit"
           clientAuth="false" sslProtocol="TLS"/>
-->
```

2. The APR implementation, which uses the OpenSSL engine by default.

   - Specify APR connector protocol in server.xml

```
<-- Define a APR SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector protocol="org.apache.coyote.http11.Http11AprProtocol"
           port="8443" .../>
```

   - Specify engine in listener tag in server.xml

```
<Listener className="org.apache.catalina.core.AprLifecycleListener"
          SSLEngine="on" SSLRandomSeed="builtin" />
```

   - Uncomment the connector in server.xml

```
<-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<!--
<Connector
           port="8443" maxThreads="200"
           scheme="https" secure="true" SSLEnabled="true"
           SSLCertificateFile="/usr/local/ssl/server.crt"
           SSLCertificateKeyFile="/usr/local/ssl/server.pem"
           clientAuth="optional" SSLProtocol="TLSv1"/>
-->
```

Note: Here we can specify the default port number(443) for https instead of 8443

6 Tomcat and LDAP

To use LDAP for authentication with Tomcat, we need to change in server.xml file of Tomcat the following:

```
<Realm className="org.apache.catalina.realm.JNDIRealm" debug="99"
connectionURL="ldap://localhost:389"
alternateURL="ldap://localhost:389"
userRoleName="member"
userBase="ou=People,dc=dcisonline,dc=uohyd=dc=ernet=dc=in"
userPattern="cn=0,cn=Users,ou=Roles,dc=dcisonline,dc=uohyd=dc=ernet=dc=in"
roleBase="cn=Users,ou=Roles,dc=dcisonline,dc=uohyd=dc=ernet=dc=in"
roleName="cn"
roleSearch="(member=0)"
roleSubtree="false"
userSubtree="true"
/>
```

Now define the role in the tomcat-users.xml and the web.xml.
Edit web.xml as follows:

```
    <security-constraint>
<display-name>Security Constraint</display-name>
<web-resource-collection>
<web-resource-name>Protected Area</web-resource-name>
<!– Define the context-relative URL(s) to be protected –>
<url-pattern>/*</url-pattern>

    <!– If you list http methods, only those methods are protected –>
</web-resource-collection>
<auth-constraint>
<!– Anyone with one of the listed roles may access this area –>
<role-name>faculty</role-name>
<role-name>student</role-name>
<role-name>staff</role-name>
<role-name>admin</role-name>
</auth-constraint>
</security-constraint>

    <!– Default login configuration uses form-based authentication –>
<login-config>
<auth-method>FORM</auth-method>
<realm-name>Form-Based Authentication Area</realm-name>
<form-login-config>
<form-login-page>index.jsp</form-login-page>
<form-error-page>error.jsp</form-error-page>
</form-login-config>
</login-config>

    <!– Security roles referenced by this web application –>
<security-role>
<role-name>faculty</role-name>
<role-name>student</role-name>
<role-name>staff</role-name>
<role-name>admin</role-name>
</security-role>
```