IBM DB2 authentication with OpenLDAP

in system landscapes like SAP

IBM Deutschland Research & Development GmbH
SAP DB2 Development Team

Author: Hinnerk Gildhoff - hinnerk@de.ibm.com
Co-Author**:** Sergiy Malikov - smalikov@de.ibm.com

1. Abstract

This document describes how to integrate DB2 and OpenLDAP in different environments with SAP applications as an example for a general system landscape. It demonstrates the configuration of an LDAP server and the appropriate LDAP clients, together with some useful hints regarding the use of common LDAP tools, and how you can adapt your SAP systems to use the benefits of OpenLDAP.

The configuration of DB2 and especially the interface between OpenLDAP and DB2 will be discussed. After you become familiar with the interaction between DB2 and OpenLDAP, the system environment will be extended to an SAP system landscape using DB2 and OpenLDAP, which has some special requirements and can have different solutions.

In this whitepaper, you will gain knowledge about setting up OpenLDAP together with DB2 in different system environments like a common SAP system landscape.

2. Table of Contents

3. Disclaimer & Trademarks

4. Introduction

LDAP is a Lightweight Directory Access Protocol that describes the communication between LDAP clients and a directory server. It is based on a client/server model which is generally used together with directory services. The directory server contains object related information which can be read by different LDAP clients with LDAP queries that fulfill the standard LDAP implementation.

One of the important reasons for using LDAP is the centralized management of logon credentials, which can be queried by any applications supporting the open LDAP standard. It's easy to use and to integrate in your existing environment. Additionally, the LDAP protocol provides great flexibility. With LDAP you can create a single point of administration across a heterogeneous environment.

The information can be general (about a company and its structure) or very detailed (about every employee in your organization or other interesting facts). You can store any kind of information in a directory server which is accessible for all clients that support the LDAP protocol.

A good example for a directory server is an address book which can be used by an email client: an email client starts for example a search request for the colleague *Frank* to find his email address. This is done with an LDAP request from the client followed by an LDAP reply from the LDAP server including the valid email address of *Frank* (see Figure 1 - LDAP Communication).

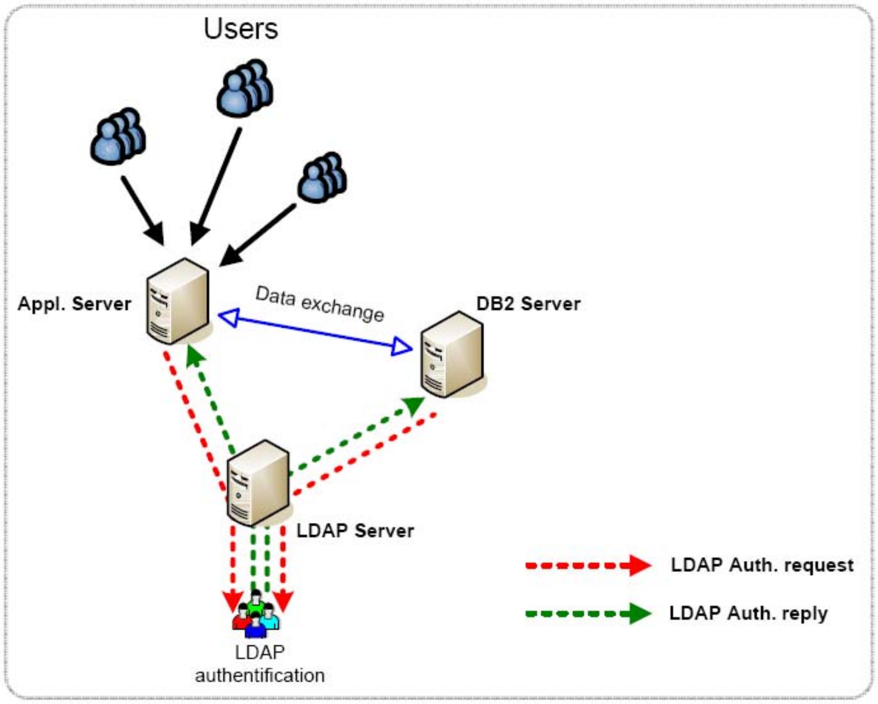


Figure 1 - LDAP Communication

Using LDAP, you are able to store information in a directory service and perform queries to this directory server. LDAP is a protocol that provides access to the X.500 directory over a TCP or SSL connection. LDAP reduces required system resources, by including only a functional subset of the original X.500 Directory Access Protocol (DAP). If you use X.500 directories and LDAP, any LDAP-enabled application can store information in the directories once (such as user authentication information) and other applications using the LDAP server will recognize it.

All entries in a directory server are arranged in a tree called the *Directory Information Tree* (DIT) and are identified through their *Distinguished Name* (DN). Each *entry* (also called *object*) in this tree belongs to one or more *object classes* and contains LDAP schemas.
The LDAP *schema* specifies the object classes and names used for the entries in the LDAP tree together with the names of the attributes and the types of operations supported by each attribute. So the structure of your LDAP tree will be based on a schema.
An *object class* describes the content and purpose of the *object*, like a class describes their instances in an object oriented programming language. Each *object* in the LDAP tree is composed of one or more *object classes*. Every *object class* has attributes stored in name value pairs. An example of an LDAP "attribute name – attribute value" pair is `uid=c0001.` Figure 2 - Simple LDAP Directory Information Tree - gives you an idea about the tree structure used by LDAP.

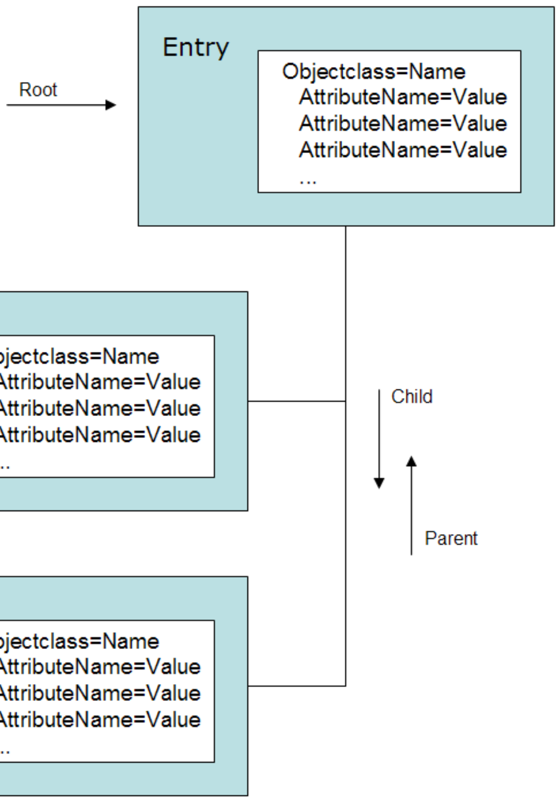


Figure 2 - Simple LDAP Directory Information Tree

LDAP protocol is extremely flexible. You can choose between different LDAP configurations for your network, like a single LDAP server or multiple LDAP servers which are accessed for different kinds of requests. For example, requests can come in from two different IP addresses, and the Web server could contact a different LDAP server for each request (in this case, the server can perform load balancing).

LDAP is used to store relatively static information. The phrase "write once read many times" describes the best use of LDAP.

LDAP is structured as a directory optimized for lookups. Its tree structure is useful for conceptualizing organizational structures.

In the following paper, we will give you an introduction about how you can use IBM® DB2® for Linux®, UNIX® and Windows® (called DB2 in the following) together with LDAP for your system landscape like SAP.

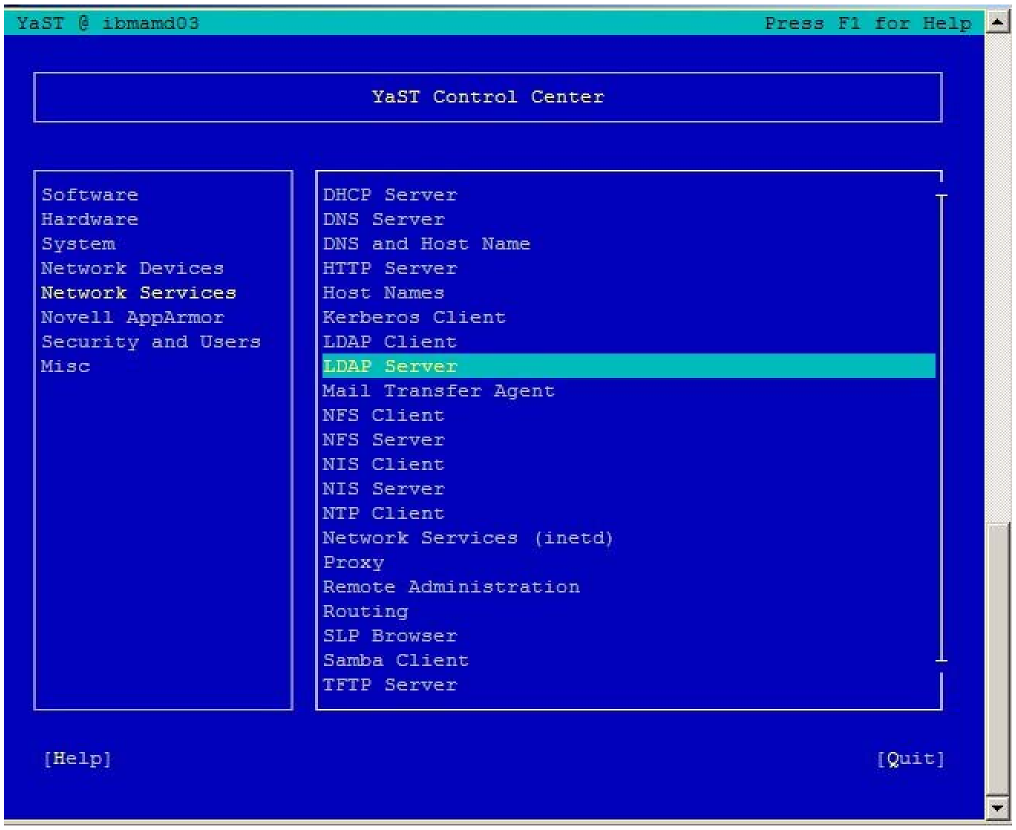
In particular, we will focus on the OpenLDAP implementation of LDAP. You will learn how to configure OpenLDAP clients and servers for usage with DB2 and how you can validate, debug and modify your configuration and optimize it that it best fits your individual needs.

5. Configuration Of OpenLDAP

In this section, we want to discuss the configuration of an OpenLDAP server together with OpenLDAP clients. You will learn how to configure OpenLDAP by using YaST (Yet another Setup Tool) for SUSE Linux, or do it directly by editing the appropriate configuration files.

5.1 OpenLDAP Server Configuration

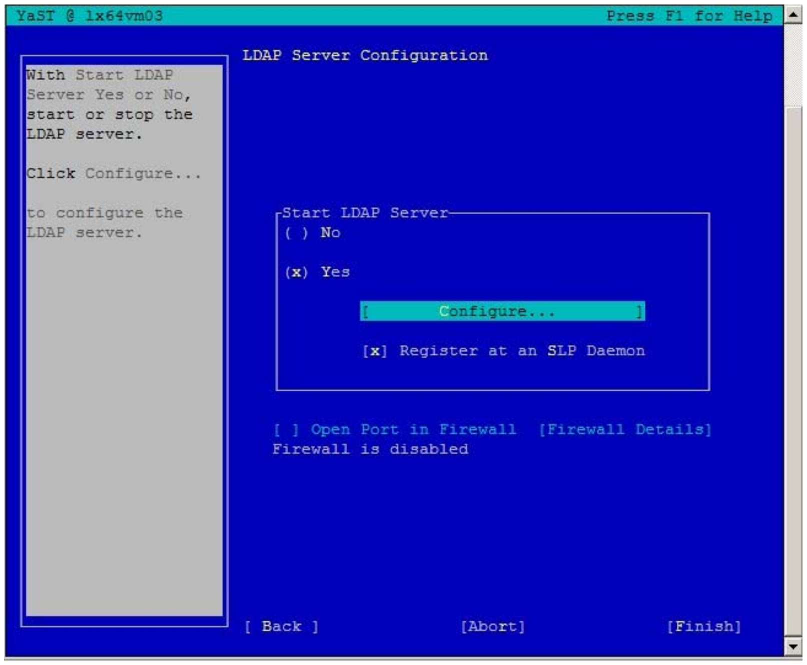
You need the *root* authority to configure your LDAP server. So login as user *root* on your host and start the installation tool *YaST* by entering the command "`yast`" in the command line. Go to *Network Services* and select *LDAP Server*. This configuration requires the *openldap2* package. If this package is not installed, you will get a prompt to do so.



```
YaST @ ibmamd03                                          Press F1 for Help

                          YaST Control Center

 Software              DHCP Server
 Hardware              DNS Server
 System                DNS and Host Name
 Network Devices       HTTP Server
 Network Services      Host Names
 Novell AppArmor       Kerberos Client
 Security and Users    LDAP Client
 Misc                  LDAP Server
                       Mail Transfer Agent
                       NFS Client
                       NFS Server
                       NIS Client
                       NIS Server
                       NTP Client
                       Network Services (inetd)
                       Proxy
                       Remote Administration
                       Routing
                       SLP Browser
                       Samba Client
                       TFTP Server

 [Help]                                                        [Quit]
```

Figure 3 - LDAP Server

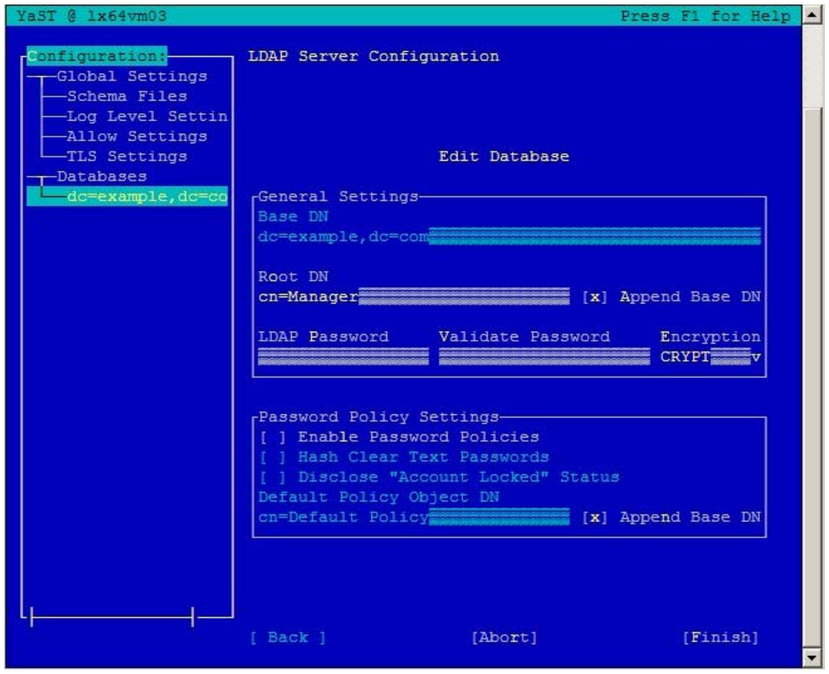
Choose *Yes* to start the LDAP server but do not finish the configuration yet! First of all, you have to configure your LDAP server. So choose *Configure* …



```
YaST @ lx64vm03                                        Press F1 for Help

                       LDAP Server Configuration
With Start LDAP
Server Yes or No,
start or stop the
LDAP server.

Click Configure...

to configure the         Start LDAP Server
LDAP server.             ( ) No

                         (x) Yes

                                [       Configure...       ]

                                [x] Register at an SLP Daemon

                        [ ] Open Port in Firewall  [Firewall Details]
                        Firewall is disabled

                       [ Back ]              [Abort]              [Finish]
```

Figure 4 - LDAP Server Configuration

The following configuration screen appears:



```
YaST @ lx64vm03                                        Press F1 for Help
Configuration:         LDAP Server Configuration
 ┬─Global Settings
 │ ├─Schema Files
 │ ├─Log Level Settin
 │ ├─Allow Settings
 │ └─TLS Settings                 Edit Database
 ┬─Databases
 └──dc=example,dc=co    General Settings
                        Base DN
                        dc=example,dc=com
                        Root DN
                        cn=Manager                  [x] Append Base DN
                        LDAP Password    Validate Password    Encryption
                                                              CRYPT    v

                        Password Policy Settings
                        [ ] Enable Password Policies
                        [ ] Hash Clear Text Passwords
                        [ ] Disclose "Account Locked" Status
                        Default Policy Object DN
                        cn=Default Policy           [x] Append Base DN

                        [ Back ]            [Abort]            [Finish]
```

Figure 5 - LDAP Server Configuration Example

Enter the following values to add a new database to your LDAP server:

Base DN = ‘*dc=example,dc=com*’
Root DN = ‘*cn=Manager*’
Password = ‘*secret*’ (retype password)
Encryption = ‘*crypt*’

An LDAP directory is like a tree structure and every LDAP directory has a top level entry which is called *Base DN*. The abbreviation *DN* stands for *distinguished name* and is very common in an LDAP context.
The *Root DN* entry is the distinct name for a user who has full access control on the LDAP directory. In this sample configuration his password is *secret* and this will be stored encrypted for security reasons.

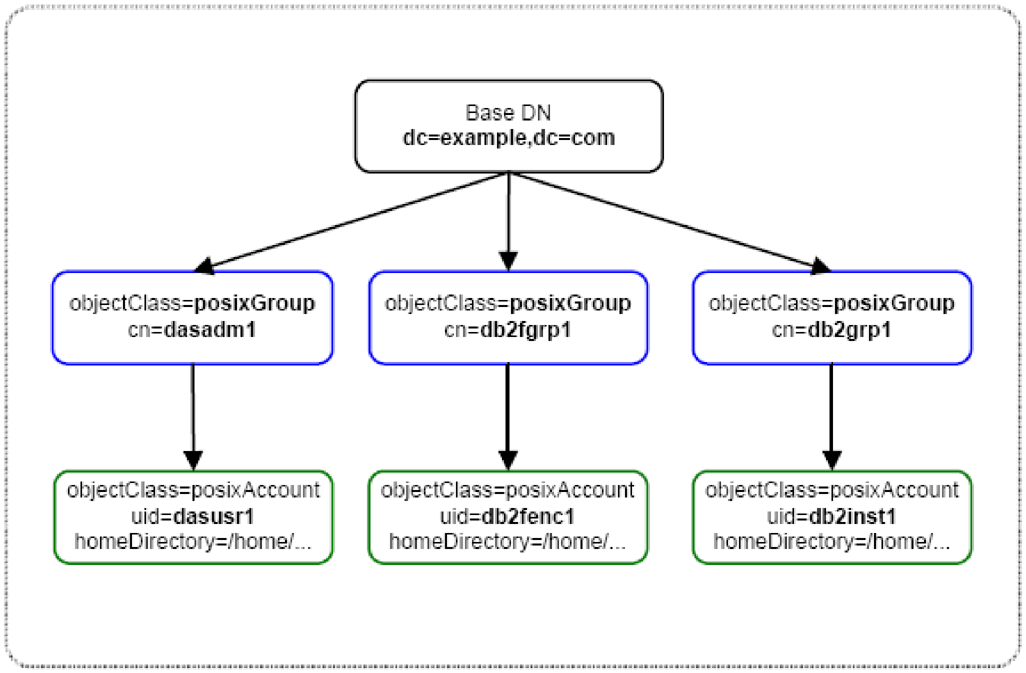


Figure 6 - LDAP Directory Structure

After you finished the *YaST* configuration, the file */etc/openldap/slapd.conf* will be updated. This file contains the configuration for the standalone server called *slapd* which handles all LDAP client requests. *Slapd* is a daemon process which is running in the background and should now also run on your system. You can check this by executing a *ps* command on unix and *grep* for *slapd*.

```
# ps -ef | grep slapd
```

The configuration file (`/etc/openldap/slapd.conf`) of the LDAP server should look similar to the following:

```
# File: /etc/openldap/slapd.conf
# At the beginning we got some general sections like
# includes, modules and security restrictions.
# We will just show the interesting part for us at the
# end of the file. The database definition:
#######################################################
# Definitions for the sample database “BDB”:
#######################################################
loglevel 0
database bdb
suffix "dc=example,dc=com"
rootdn "cn=Manager,dc=example,dc=com"
rootpw "{crypt}IFJhmeMCyi0TU"
directory /var/lib/ldap/
checkpoint 1024 5
cachesize 10000
```

All LDAP entries will be stored in the database called *bdb* which is located on the filesystem in the directory */var/lib/ldap*. All entries will be saved under the root object *dc=example,dc=com* as you specified it before and only the user *rootdn* has full access to these objects by using the root password *secret*.

5.2 OpenLDAP Client Configuration

The installation tool *YaST* supports also the configuration of LDAP clients. So start *YaST* again and go to *Network Services* and select *LDAP client*.

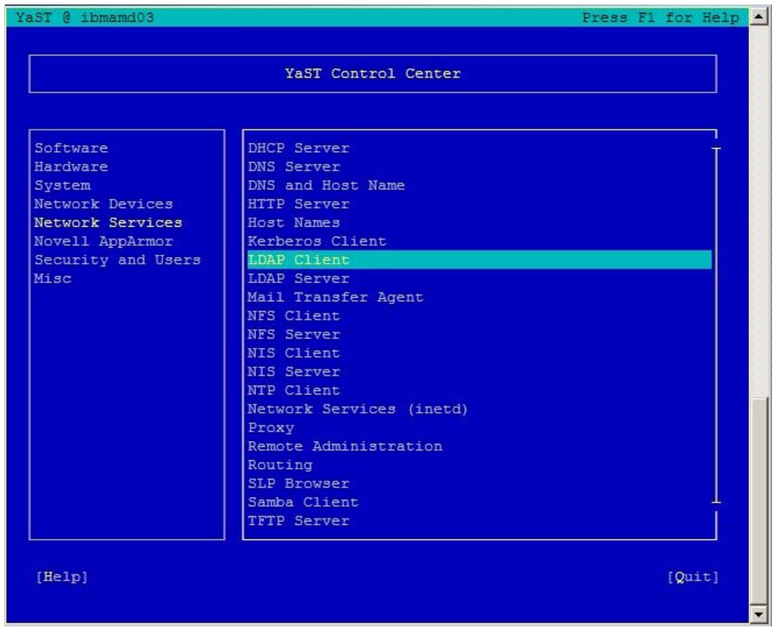
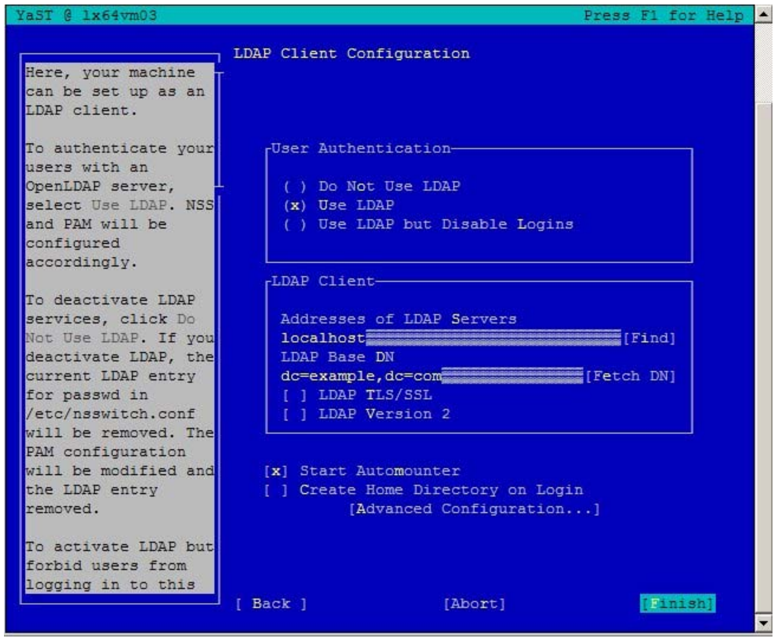

```
YaST @ ibmamd03                                   Press F1 for Help

                        YaST Control Center

Software               DHCP Server
Hardware               DNS Server
System                 DNS and Host Name
Network Devices        HTTP Server
Network Services       Host Names
Novell AppArmor        Kerberos Client
Security and Users     LDAP Client
Misc                   LDAP Server
                       Mail Transfer Agent
                       NFS Client
                       NFS Server
                       NIS Client
                       NIS Server
                       NTP Client
                       Network Services (inetd)
                       Proxy
                       Remote Administration
                       Routing
                       SLP Browser
                       Samba Client
                       TFTP Server

[Help]                                                  [Quit]
```

Figure 7 - LDAP Client

The following *LDAP Client Configuration* dialog appears:



```
YaST @ lx64vm03                                              Press F1 for Help

                         LDAP Client Configuration
Here, your machine
can be set up as an
LDAP client.

To authenticate your      User Authentication
users with an
OpenLDAP server,            ( ) Do Not Use LDAP
select Use LDAP. NSS        (x) Use LDAP
and PAM will be             ( ) Use LDAP but Disable Logins
configured
accordingly.
                          LDAP Client
To deactivate LDAP
services, click Do          Addresses of LDAP Servers
Not Use LDAP. If you        localhost                         [Find]
deactivate LDAP, the        LDAP Base DN
current LDAP entry          dc=example,dc=com                 [Fetch DN]
for passwd in               [ ] LDAP TLS/SSL
/etc/nsswitch.conf          [ ] LDAP Version 2
will be removed. The
PAM configuration
will be modified and      [x] Start Automounter
the LDAP entry            [ ] Create Home Directory on Login
removed.                          [Advanced Configuration...]

To activate LDAP but
forbid users from
logging in to this
                          [ Back ]            [Abort]            [Finish]
```

Figure 8 - LDAP Client Configuration

First of all, set the user authentication to '*Use LDAP*'. The next value is the address of your LDAP Server. Here you can enter the IP address or name of your server where the LDAP deamon is running or you choose *localhost* if you are still on the same server as above. The LDAP *Base DN* needs to be the same as the one you configured for your LDAP server. So in this case, we take the *Base DN* '*dc=example,dc=com*'.

Note that we got our first design decision here. You can install your LDAP server on a dedicated server and the LDAP clients on different machines or you install everything on the same host. If you are a beginner with LDAP and DB2 and you just want to learn and test a little bit around, you can install everything on the same server, but in all other cases we recommend to choose a dedicated LDAP server for your environment, because the LDAP server should be a central point in your system landscape which has to answer many LDAP requests and so it's better to use dedicated hardware resources for this specific task only.

If you save the OpenLDAP client configuration, you will get a popup message that these changes affect only newly created processes and not already running services. In our situation you could easily ignore the warning because we only start new processes and *sshd* will be restarted automatically by YaST which is important for remote logins. So select "*OK*" , and YaST will save the configuration and will directly start the client on your local machine.

The OpenLDAP **client** configuration will be saved in the file *ldap.conf*. Usually there are two different *ldap.conf* files: one with the common configuration, and another file with the specific configuration parameters.
You will find all your changes in */etc/ldap.conf* which is the common configuration file for the LDAP client. The other file located in */etc/openldap/ldap.conf* is a special configuration for all our OpenLDAP command line tools which we will discuss later.

The common LDAP client configuration file should contain the following entries:

```
# File: /etc/ldap.conf
# The file contains lots of more entries and many of them
# are comments. We show only the interesting values for now

host                localhost
base                dc=example,dc=com
ldap_version        3

pam_password        crypt
pam_filter          objectclass=posixAccount

nss_map_attribute uniqueMember member
nss_base_passwd     dc=example,dc=com
nss_base_shadow     dc=example,dc=com
nss_base_group      dc=example,dc=com
```

The LDAP client queries the LDAP server at the specified *host* which could also be an IP address. The client starts all queries at the root object *dc=example,dc=com* which is also known as *base DN* and uses the LDAP version 3 (LDAPv3).
PAM (Pluggable Authentication Module) is an API for authentication services. We use this common interface for LDAP authentication together with an encrypted password and a special LDAP object of type *posixAccount*. All LDAP objects of this type represent an abstraction of an account with POSIX (Portable Operating System Interface) attributes.
NSS (Network Security Services) is a set of libraries to support cross-platform development of security-enabled client and server applications. This includes crypto libraries like SSL, TLS, PKCS S/MIME and other security standards. We have to specify the *base DN* for this interface again and additionally two mapping attributes that we will discuss later.

The specialized OpenLDAP client configuration file contains fewer entries:

```
# File: /etc/openldap/ldap.conf

host    localhost
base    dc=example,dc=com
```

We just define the host of the LDAP server and the *base DN*. This is enough for the *OpenLDAP* command line tools to work.

5.3 OpenLDAP Environment Validation

After you configured your LDAP server and at least one LDAP client, you should validate your configurations to prevent errors that could occur later in a complex environment. It is always a good idea to validate and test your configuration in an early state so we can find possible problems in a manageable environment and do not have to investigate a lot of time in troubleshooting at the end of our project.

We will validate your current LDAP environment with some useful system commands and by using the *OpenLDAP* client tools which are easily to find because they always use the prefix *ldap* in their name. So just type *ldap* in your bash command line interpreter and hit the tab-key twice to list all installed *OpenLDAP* tools.

5.3.1 Check if your local OpenLDAP server is running

1. First of all, check if your local LDAP server is running. You can do this by executing the following command:

```
# ps -ef | grep -i ldap
```

This command should list the LDAP deamon which represents your LDAP server:

```
/usr/lib/openldap/slapd -h ldap:/// -u ldap -g ldap -o slp=on
```

This indicates that your LDAP server (daemon) is running and is waiting for incoming requests.

Note:
If there is no such process, you can start the LDAP server with the ***rcldap*** command. Just type the following command to start the LDAP server:

```
# rcldap start
```

If you restart the LDAP server, it will take a while till the server works again.
You can monitor this in the file */var/log/messages* by issuing the following command:

```
# tail -f /var/log/messages
```

5.3.2 Test your connection to the LDAP server with *ldapsearch*

The *ldapsearch* command opens a connection to an LDAP server, binds to it and performs a search query which can be specified by using special parameters.
You could also use this search command to validate if you get a connection to your LDAP sever. Use the following command to connect to your LDAP server with a simple authentication mechanism by using the *–x* parameter instead of a more complex mechanism like SASL (Simple Authentication and Security Layer):

```
# ldapsearch –x
```

Your LDAP server should reply with a response containing all of your LDAP entries in an LDIF (LDAP Data Interchange Format) like syntax. This ensures that your LDAP server is running and you can connect and bind to it.
You get all LDAP entries because the command uses the default search filter *(objectClass=*).*

The response of your LDAP server should look similar to the following:

```
# extended LDIF
#
# LDAPv3
# base <> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
# example.com
dn: dc=example,dc=com
dc: example
o: example
objectClass: organization
objectClass: dcObject

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

The search request should find one entry in the directory information tree (numEntries: 1). You have configured your server with the Base DN *dc=example,dc=com*. This object was already created and stored in the LDAP directory for you by the YaST installation process.

Note:
The last parameter –x is used for a simple authentication instead of SASL. The SASL framework is described in detail in RFC2222. It offers many different authentication mechanisms like DIGEST-MD5 and Kerberos V to provide secure authentication services. The standard client tools provided with OpenLDAP Software, such as ldapsearch(1) and ldapmodify(1), will by default attempt to authenticate the user on the slapd(8) server using SASL. We use the –x parameter to get a simple connection to the LDAP server.

5.3.3 Clean up your OpenLDAP directory with *ldapdelete*

Now, we want to delete the sample root object and create our own structure manually from the beginning. Issue the following command to delete the existing LDAP entry:

```
# ldapdelete –x  “dc=example,dc=com”
```

The command should lead to an error message like the following:

```
ldap_delete: Strong(er) authentication required(8)
   additional_info: modifications require authentication
```

This happens because this time we are not only reading the directory. We want to modify (delete) contents of the directory server. Therefore, we need more privileges, and this requires a stronger authentication. This can be done by specifying the additional parameter *–D* for a distinguished bind name and *–w* parameter to specify the password for this user.

The specified user should be the same user as defined in */etc/openldap/slapd.conf.*
So we need the *rootdn* user together with the *rootpw* which is stored encrypted in the configuration file. You can either user the *–w* option to specify the password directly in the system call or use *–W* to enforce a prompt for the password. In this chapter we use *–w* for a better demonstration, but note that the –W option is more secure and therefore should be used in production systems.

Execute the following command:

```
# ldapdelete -v
             –x
             –D "cn=Manager,dc=example,dc=com"
```

```
                    -W
                    “dc=example,dc=com”
```

Notes:
The option *-v* switches the verbose mode on.
The distinguished name (“-D” option) must be the fully specified name, because only the whole string is guaranteed to be unique. The last parameter *“dc=example,dc=com”* specifies the object to delete.

Verify if your directory information tree is empty by issuing a new
“ `ldapsearch -x` ” command. This time you should not find any entries.

If you want to know more about the LDAP commands, you can use the documentation links at the end of this Whitepaper, or Linux man pages:

```
  # man ldapmodify
```

6. Configuration of DB2

After ensuring that our LDAP server and clients are working, we want to configure our DB2 database for use with LDAP. We will discuss how we can install and configure our database to use our LDAP environment for the DB2 user authentication process. Note that you can use a dedicated database server or the same server where your LDAP server is running. We will discuss the differences in these two approaches later again.

6.1 Configure DB2 and LDAP Interaction Plug-ins

IBM provides a free package with LDAP plug-ins for DB2 (requires at least DB2 UDB Version 8.2). This package includes three DB2 security plug-ins: one for server side authentication, one for client side authentication and the last one for group lookup. Depending on your specific environment, you may need to use one, two or all three plug-ins.

For DB2 version 9.5 and newer, these LDAP plug-ins are already included in the default installation of the DB2 product.

These plug-ins do not support environments where some users are defined in LDAP and others in the operating systems. If you decide to use the LDAP plug-ins, you have to define all users associated with the database in the LDAP server. The same principle applies to the group plug-in.

First of all, you have to decide which plug-ins are mandatory for your system landscape. In most cases you can **skip** the client plug-in. The client authentication plug-in is used in scenarios where the user ID and the password validation supplied on a CONNECT or ATTACH statement occurs on the client system. So the database manager configuration parameters SRVCON_AUTH or AUTHENTICATION need to be set to the value CLIENT. However, the CLIENT authentication is difficult to secure and is not generally recommended.

The **server** plug-in is generally recommended because it performs a server side validation of user IDs and passwords if clients execute a CONNECT or ATTACH statement and this is definitely the secure way. Note, that the server plug-in also provides a way to map LDAP user IDs to DB2 authorization IDs.

The group lookup plug-in retrieves group membership information from the LDAP server for a particular user. It is generally required if you wish to use LDAP to store your group definitions. As long as you are using DB2, you should use the group plug-in, because DB2 uses an indirect authentication process as we will see later in this chapter.

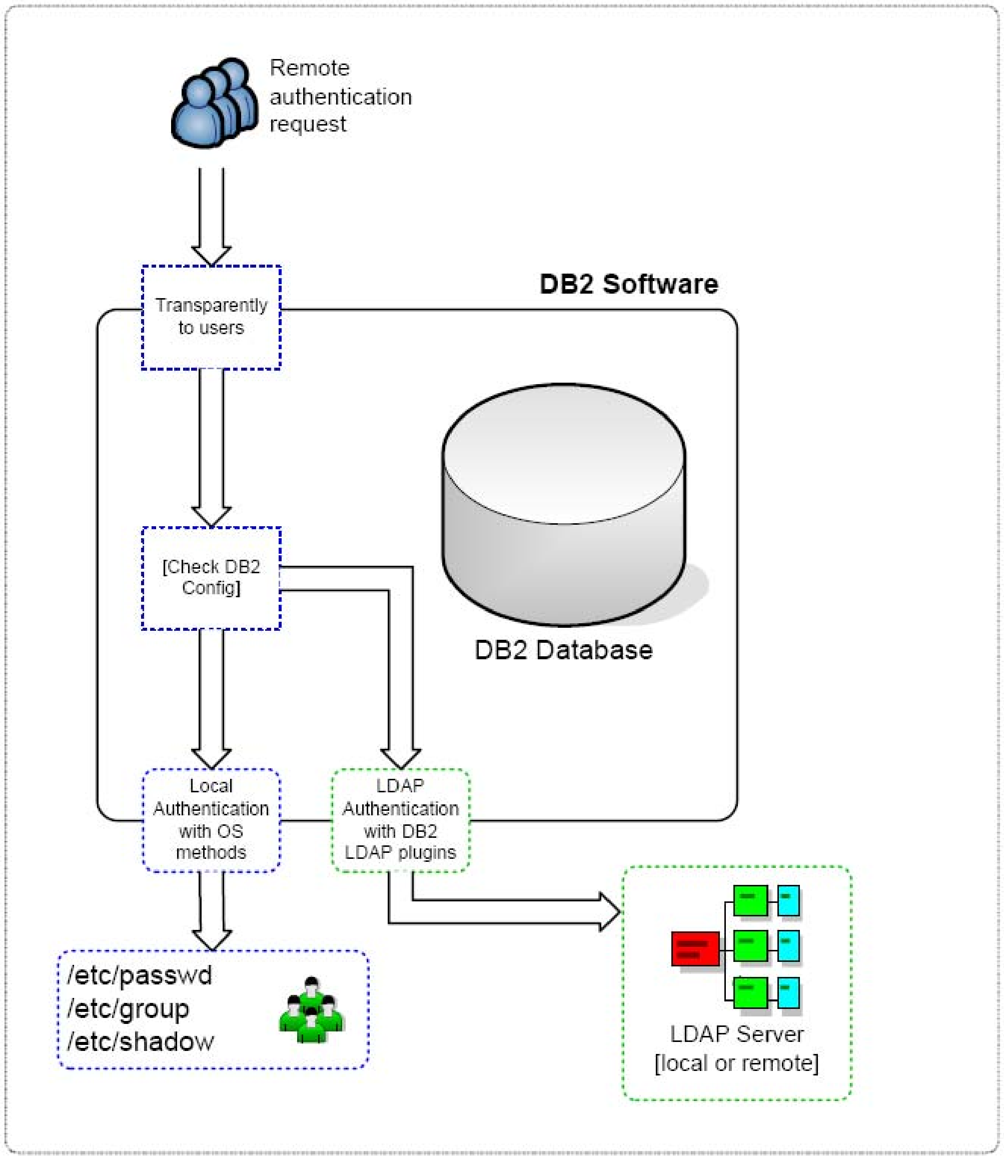

Remote authentication request
DB2 Software
Transparently to users
[Check DB2 Config]
DB2 Database
Local Authentication with OS methods
LDAP Authentication with DB2 LDAP plugins
/etc/passwd
/etc/group
/etc/shadow
LDAP Server [local or remote]

Figure 9 - Plug-In Interaction Diagram

Before we can start with the installation and configuration of the DB2 security plug-ins, we need to think about the required directory information tree for DB2. Most of the time, DB2 uses indirect authorization which means that a user belongs to a group and this group was granted with some authorities (this default behavior could be overwritten). This indicates that we should not only define the DB2 users in the LDAP server, but instead, we need to define all DB2 users and all DB2 groups in the LDAP directory too. So for a default DB2 database, the structure has to look like this:

```
            example.com --+
                          |
                          +-- dasadm1  --+-- dasusr1
                          |
                          +-- db2grp1  --+-- db2inst1
                          |
                          +-- db2fgrp1 --+-- db2fenc1
```

Figure 10 - DB2 DIT Example

The corresponding LDIF file should look like this (note that we added the root object, because we deleted it before):

```
#
# LDAP root object
# example.com
#
dn: dc=example,dc=com
dc: example
o: example
objectClass: organization
objectClass: dcObject

#
# db2 groups
#


dn:          cn=dasadm1,dc=example,dc=com
cn:          dasadm1
objectClass: top
objectClass: posixGroup
gidNumber:   300
objectClass: groupOfNames
member:      uid=dasusr1,cn=dasadm1,dc=example,dc=com
memberUid:   dasusr1



dn:          cn=db2grp1,dc=example,dc=com
cn:          db2grp1
objectClass: top
objectClass: posixGroup
gidNumber:   301
objectClass: groupOfNames
member:      uid=db2inst1,cn=db2grp1,dc=example,dc=com
memberUid:   db2inst1


dn:          cn=db2fgrp1,dc=example,dc=com
cn:          db2fgrp1
objectClass: top
```

```
objectClass: posixGroup
gidNumber:   302
objectClass: groupOfNames
member:      uid=db2fenc1,cn=db2fgrp1,dc=example,dc=com
memberUid:   db2fenc1



#
# db2 users
#

dn:             uid=dasusr1,cn=dasadm1,dc=example,dc=com
cn:             dasusr1
sn:             dasusr1
uid:            dasusr1
objectClass:    top
objectClass:    inetOrgPerson
objectClass:    posixAccount
uidNumber:      300
gidNumber:      300
loginShell:     /bin/bash
homeDirectory: /home/dasusr1


dn:            uid=db2inst1,cn=db2grp1,dc=example,dc=com
cn:            db2inst1
sn:            db2inst1
uid:           db2inst1
objectClass:   top
objectClass:   inetOrgPerson
objectClass:   posixAccount
uidNumber:     301
gidNumber:     301
loginShell:     /bin/bash
homeDirectory: /home/db2inst1



dn:             uid=db2fenc1,cn=db2fgrp1,dc=example,dc=com
cn:             db2fenc1
sn:             db2fenc1
uid:            db2fenc1
objectClass:    top
objectClass:    inetOrgPerson
objectClass:    posixAccount
uidNumber:      303
gidNumber:      303
loginShell:     /bin/bash
homeDirectory: /home/db2fenc1
```

Figure 11 - DB2 LDIF Example

Use the above LDIF example and paste it in a file called *db2.ldif* in your file system. We use this file to add the defined structures to your LDAP directory.

To add the DB2 users and DB2 groups to the directory information tree, we need to bind the user *rootdn* to the LDAP server in order to get the right privileges. Execute the following *ldapadd* command to fill the LDAP information tree with all our objects defined in the LDIF file *db2.ldif*:

```
# ldapadd -x
          -D “cn=Manager,dc=example,dc=com”
          -W
          -f <path>/db2.ldif
```

Perform a search request in your LDAP directory and verify that all objects are listed in your LDAP directory. Pipe the long output to *more* which is a filter for paging through text:

```
# ldapsearch -x | more
```

The result set should contain seven objects as you can see in Figure 11 - DB2 LDIF Example.

In case you have to search in a very complex information tree, you need to know more about search filters. For example, if you want to find all DB2 users only, you can do this by using the following search query:

```
# ldapsearch -x “(&(uid=*)(|(ou=db2*)(ou=das*))”
```

This search query uses a search filter that finds all objects with the attribute *uid* containing a value starting with *das* or *db2*. If you want to get more information about search filters for LDAP, take a look at the Backus-Naur-Form (BNF) defined in RFC1558.
The search query should find three objects that fulfill the search criteria: one result for every DB2 user.

6.2 Prepare file system for DB2 usage

In this step we want to install the DB2 software and create an instance for our LDAP user *db2inst1*. This requires an existing home directory for the user *db2inst1* with two empty files inside which will be modified by DB2 with some environment settings.

If not already done, install the DB2 software. We recommend using the following root command to install the DB2 software only, without creating any instances or database users:

```
# <db2_source>/ESE/disk1/db2_install
```

After the installation is finished successfully, prepare your file system for the next tasks:

Before you can create a DB2 instance, create the home directory for the database administrator *db2inst1*, who will be the owner of the database instance. His home directory is used to store all DB2 instance data. Additionally, you should also create the files *.profile* and *.login* in his home directory which will be modified by DB2 as well. DB2 will add some lines to these files to ensure that the instance owner has the appropriate environment required for the DB2 command line. Execute the following commands:

```
# mkdir /home/db2inst1

# > /home/db2inst1/.login

# > /home/db2inst1/.profile
```

Since we have registered all our DB2 users in the LDAP directory, we can create the instance *db2inst1* with the instance owner id *db2inst1* and use the fenced user id *db2fenc1*, which is needed for running user defined functions (UDFs) or stored procedures.

```
# /opt/ibm/db2/V9.5/instance/db2icrt –u db2fenc1 db2inst1
DBI1070I Program db2icrt completed successfully.
```

Take a look at the home directory structure and note that we got a new sub-directory called *sqllib* where all instance information is stored. Furthermore, the profile- and login- files have been customized for the usage with DB2:

```
# ls –la /home/db2inst1
.login
.profile
sqllib
```

6.3 Configure authentication plug-ins for LDAP support in DB2

In this section, we will configure DB2 to use the server plug-in for a server side authentication and the group plug-in for group lookups within the LDAP directory.

For DB2 versions older than 9.5, you have to start with the installation of the server and the group plug-in. You can skip this part and jump directly to the plug-in configuration part if you have DB2 version 9.5 or newer, since the plug-ins are already included with these DB2 versions.

You can download the LDAP plug-ins for DB2 version 8.2 or 9.1 from the following site: https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?lang=en_US&source=swg-dm-db2ldap

Copy the required LDAP plug-ins (shared object files) to the appropriate DB2 directory:

```
# cp /<db2_ldap_pkg>/<os>/v9/IBMLDAPauthserver.so
     /home/db2inst1/sqllib/security<bit>/plugin/server/.

# cp /<db2_ldap_pkg>/<os>/v9/IBMLDAPgroups.so
     /home/db2inst1/sqllib/security<bit>/plugin/group/.
```

Once the plug-ins have been copied to the specified directory, you have to switch to the DB2 instance owner and change the database manager configuration to use these plug-ins. Additionally, configure the database manager to use server side encrypted authentication.

```
# su – db2inst1

db2inst1> db2 update dbm cfg using
              srvcon_pw_plugin IBMLDAPauthserver

db2inst1> db2 update dbm cfg using
              group_plugin IBMLDAPgroups

db2inst1> db2 update dbm cfg using
              authentication SERVER_ENCRYPT

db2inst1> exit
```

These parameter changes have a deferred effect. They will only be active if you restart your instance.
However, before you do this, you have to install and configure the main DB2-LDAP configuration file called *IBMLDAPSecurity.ini* so that the DB2 plug-ins work will be able to work with the current LDAP configuration.

The *IBMLDAPSecurity.ini* file may contain the following sample configuration:

```
;-----------------------------------------------------------
; SERVER RELATED VALUES
;-----------------------------------------------------------

; Name of your LDAP server(s).
; This is a space separated list of LDAP server addresses,
; with an optional port number for each one:
;    host1[:port] [host2:[port2] ... ]
; The default port number is 389, or 636 if SSL is enabled.
LDAP_HOST = my.ldap.server

;-----------------------------------------------------------
; USER RELATED VALUES
;-----------------------------------------------------------

; LDAP object class used for users
USER_OBJECTCLASS = posixAccount

; LDAP user attribute that represents the "userid"
; This attribute is combined with the USER_OBJECTCLASS and
; USER_BASEDN (if specified) to construct an LDAP search
; filter when a user issues a DB2 CONNECT statement with an
; unqualified userid. For example, using the default values
; in this configuration file, (db2 connect to MYDB user bob
; using bobpass) results in the following search filter:
;    &(objectClass=inetOrgPerson)(uid=bob)
USERID_ATTRIBUTE = uid

; LDAP user attribute, representing the DB2 authorization ID
AUTHID_ATTRIBUTE = uid



;-----------------------------------------------------------
; GROUP RELATED VALUES
;-----------------------------------------------------------

; LDAP object class used for groups
GROUP_OBJECTCLASS = groupOfNames

; LDAP group attribute that represents the name of the group
GROUPNAME_ATTRIBUTE = cn


; Determines the method used to find the group memberships
; for a user. Possible values are:
;  SEARCH_BY_DN   - Search for groups that list the user as
;                   a member. Membership is indicated by the
;                   group attribute defined as
;                   GROUP_LOOKUP_ATTRIBUTE.
;  USER_ATTRIBUTE - A user's groups are listed as attributes
;                   of the user object itself. Search for the
;                   user attribute defined as
;                   GROUP_LOOKUP_ATTRIBUTE to get the groups.
GROUP_LOOKUP_METHOD = SEARCH_BY_DN

; GROUP_LOOKUP_ATTRIBUTE
; Name of the attribute used to determine group membership,
; as described above.
; GROUP_LOOKUP_ATTRIBUTE  = ibm-allGroups
GROUP_LOOKUP_ATTRIBUTE  = member
```

Copy the sample IBM LDAP configuration file to its intended location:

```
# cp /iod/db2_ldap_pkg/IBMLDAPSecurity.ini
     /home/db2inst1/sqllib/cfg/
```

After you configured DB2 to use the installed LDAP plug-ins and the DB2-LDAP configuration file, you have to restart your DB2 instance. Since we never had a running instance, a *db2start* should be enough to activate our changes and if this LDAP user can start the instance, we are really making progress. So try to start the instance:

```
# su – db2inst1

db2inst1> db2start
```

The initial *db2start* command will lead to a security error message. This occurs because your DB2 security configuration is not yet correctly configured for your LDAP environment. Read the next section to learn how to customize your DB2-LDAP configuration file for your specific environment.

6.4 Customize DB2 and OpenLDAP configuration

Currently, DB2 is trying to authenticate all DB2 users with the recently installed security plug-ins according to the default DB2-LDAP configuration. The default configuration is not adequate for our purposes.

Therefore, we have to customize the configuration file `IBMLDAPSecurity.ini` for our specific LDAP environment. In this step, we use the LDAP environment defined in Figure 10 - DB2 DIT Example. Open the configuration file and perform the following changes:

First of all, we need to change the value of LDAP_HOST to our LDAP host which could be *localhost* or any valid IP address.

The next assignment is the value of USER_OBJECTCLASS. To fully understand this, we need to discuss first what an object class is:
LDAP entries use standardization to ensure the collaboration with other directory services, like DNS (Domain Name System), NIS (Network Information Service) and plenty of other services. This requires an LDAP schema which defines a list of possible object classes together with attributes. An entry can have more than one object class which defines all possible attributes of this entry. So everybody who knows the schema of an object knows also the attributes of the object and can use them for searching or other operations. Here you can define one object class for all LDAP user objects so that the DB2 user plug-in can use known attributes to search for his DB2 users.

For example, the object class *inetOrgPerson* contains attributes like *departmentNumber* and *initials*. As you might guess, in our configuration we do not really need the object class *inetOrgPerson* because we have DB2 user accounts and no real people to manage. Instead, we need some attributes like *gidNumber* and *homeDirectory* which is defined in the *posixAccount*. Therefore, the value *posixAccount* is a good choice for USER_OBJECTCLASS.

The USERID_ATTRIBUTE and AUTHID_ATTRIBUTE already have the correct values. Use the attribute *uid* to reference the user and the authentication id of the user (see *uid* in Figure 11 - DB2 LDIF Example).

The next configuration parameters are for group relations.

The GROUP_OBJECTCLASS uses the same mechanism as described above for the USER_OBJECTCLASS. The default value “`groupOfNames`” is correct. We use the class *groupOfNames* to establish the relation between groups and users.

The GROUPNAME_ATTRIBUTE indicates which attribute of the group entries represents the name of the group. As you might guess, this should be the *cn* attribute.

The parameter GROUP_LOOKUP_METHOD defines a method for finding the members of a group. We want to search for groups which list special users as a member. Membership is indicated by the value of the group attribute defined as GROUP_LOOK_ATTRIBUTE.

This parameter is important, because we want to find the relation between users and groups with a specific user attribute. We need to know this relation attribute for a group lookup.
Use the default value *member* for the GROUP_LOOKUP_ATTRIBUTE which represents the group membership of our DB2 users.

So the correct configuration of *IBMLDAPSecurity.ini* should look like the following configuration:

```
;------------------------------------------------------------
; SERVER RELATED VALUES
;------------------------------------------------------------
LDAP_HOST = localhost

;------------------------------------------------------------
; USER RELATED VALUES
;------------------------------------------------------------
USER_OBJECTCLASS = posixAccount

USER_BASEDN      = dc=example,dc=com

USERID_ATTRIBUTE = uid

AUTHID_ATTRIBUTE = uid

;------------------------------------------------------------
; GROUP RELATED VALUES
;------------------------------------------------------------
GROUP_OBJECTCLASS      = groupOfNames

GROUP_BASEDN           = dc=example,dc=com

GROUPNAME_ATTRIBUTE    = cn

GROUP_LOOKUP_METHOD    = SEARCH_BY_DN

GROUP_LOOKUP_ATTRIBUTE = member
```

This configuration defines that every LDAP user uses the attribute *uid* for identification, has at least the object class *posixAccount* and is part of the Base DN *dc=example,dc=com*. LDAP groups can be identified through the attribute *cn*, are also part of the Base DN dc=*example,dc=com*, and have at least the object class *groupOfNames*. The relationships between users and groups are configured through the group attribute *member* that references all members of a group. In addition, it is important to know that all user and group objects have the same Base DN.

Now, you should take another look at our *db2.ldif* file to really understand this configuration (see Figure 11 - DB2 LDIF Example) and note that there are lots of different ways to configure an LDAP environment. This is just an example.

After you changed the configuration file `IBMLDAPSecurity.ini`, the configuration will immediately take effect and your DB2 and LDAP environment should work. You will validate your modified configuration in the next step.

6.5 DB2 and LDAP environment validation (optional)

In this step we want to debug the behavior of LDAP and DB2 a little bit and take a deeper look in the cooperation of both techniques.

Before we start the validation or debugging of the LDAP behavior, we want to change the LDAP log level to get more information. Open the configuration file and set *loglevel* to *-1* which is the highest stage for logging. It simply logs all information regarding LDAP:

```
# vi /etc/openldap/slapd.conf

loglevel -1
```

Next, we need to open two shells. One for executing system commands and the other one for monitoring the system behaviour. You can do this in many ways. Start the X server and open two shells or use two Putty sessions etc..

Give every shell a unique identifier so you can better navigate between your windows.

Set the prompt for your first shell to *monitor*:

```
# export PS1='$PWD monitor> '
```

This *monitor* shell should lists the tail of the system log where we find all occurring LDAP messages. So execute the following command:

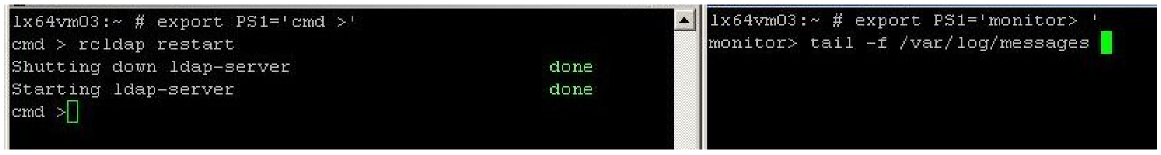
```
monitor> tail –f /var/log/messages
```

Now set the prompt of your second shell to *cmd*:

```
# export PS1='$PWD cmd> '
```

This *cmd* shell is for executing system commands. Here you should restart the LDAP server and watch the LDAP activity in the *monitor* window. The restart is required to activate the new log level. Execute the following command:

```
cmd> rcldap restart
```



```
lx64vm03:~ # export PS1='cmd >'
cmd > rcldap restart
Shutting down ldap-server                                 done
Starting ldap-server                                      done
cmd >
```

```
lx64vm03:~ # export PS1='monitor> '
monitor> tail -f /var/log/messages
```

Figure 12 - LDAP logging

Now the *monitor* starts to show you the LDAP actions. You will see something like:

```
  nss_ldap: reconnecting to LDAP server (sleeping XX seconds)…
```

This could take a while - depending on your hardware - till your LDAP server is reconnected again.

```
  nss_ldap: reconnected to LDAP server ldap://localhost
```

You can check if your LDAP server is running again, if you execute a simple search query iny your *cmd* window. It will give you an error message or it will show you the search results if it can contact the LDAP server again:

```
  cmd> ldapsearch –x
  ldap_bind: Cannot contact LDAP server (-1)
```

After your LDAP server is running again, switch to your LDAP user *db2inst1* and check if your DB2 instance is listed.

```
  cmd> su – db2inst1
  db2inst1> db2ilist
  db2inst1
```

Now create a sample database and try to connect to this database. This definitely ensures that the reference between your LDAP user and LDAP groups works, because in DB2 the important DB2 authorization rights are configured for groups and not directly for users. So you need to have the DB2 users in the appropriate Unix group if you want to create a new database.

Create the sample database:

```
 db2inst1> db2sampl
```

Check if your database is correctly configured:

```
 db2inst1> db2 list db directory

    System Database Directory

    Number of entries in the directory = 1

   Database 1 entry:

    Database alias                       = SAMPLE
    Database name                        = SAMPLE
    Local database directory             = /home/db2inst1
    Database release level               = c.00
    Comment                                    =
    Directory entry type                       = Indirect
    Catalog database partition number          = 0
    Alternate server hostname                  =
    Alternate server port number               =
```

Connect to your database with a normal connect statement and watch the activities in your *monitor* window:

```
db2inst1> db2 connect to sample

   Database Connection Information

 Database server        = DB2/LINUXX8664 9.5.1
 SQL authorization ID   = DB2INST1
 Local database alias   = SAMPLE
```

Congratulations, your connection was successful! Your LDAP user has all required privileges to connect to your database.

As long as you have an open database connection, you can check your authorizations (otherwise, you will get the error message SQL1024N). Execute the following DB2 statement to list the direct and indirect authorizations of your LDAP user *db2inst1*:

```
db2inst1> db2 get authorizations

 Administrative Authorizations for Current User

 Direct SYSADM authority                        = NO
 Direct SYSCTRL authority                       = NO
 Direct SYSMAINT authority                      = NO
 Direct DBADM authority                         = YES
 Direct CREATETAB authority                     = YES
 Direct BINDADD authority                       = YES
 Direct CONNECT authority                       = YES
 Direct CREATE_NOT_FENC authority               = YES
 Direct IMPLICIT_SCHEMA authority               = YES
 Direct LOAD authority                          = YES
 Direct QUIESCE_CONNECT authority               = YES
 Direct CREATE_EXTERNAL_ROUTINE authority       = YES
 Direct SYSMON authority                        = NO

 Indirect SYSADM authority                      = YES
 Indirect SYSCTRL authority                     = NO
 Indirect SYSMAINT authority                    = NO
 Indirect DBADM authority                       = NO
 Indirect CREATETAB authority                   = YES
 Indirect BINDADD authority                     = YES
 Indirect CONNECT authority                     = YES
 Indirect CREATE_NOT_FENC authority             = NO
 Indirect IMPLICIT_SCHEMA authority             = YES
 Indirect LOAD authority                        = NO
 Indirect QUIESCE_CONNECT authority         = NO
 Indirect CREATE_EXTERNAL_ROUTINE authority = NO
 Indirect SYSMON authority                  = NO
```

Now terminate your open connection.

```
db2inst1> db2 terminate
```

We want to try some other connection statements. All these statements should work. You just have to enter the right password for your LDAP user.

Before we try other connection statements, use the tool *ldappasswd* to generate a password for your LDAP user:

```
db2inst1> ldappasswd –x
                   -D “cn=Manager,dc=example,dc=com”
                   -W
                   "uid=db2inst1,cn=db2grp1,dc=example,dc=com"
New password: <password>
Result: Success (0)
```

Connect with a part of the user DN and type the generated password when prompted and do not forget to take a look in your monitor xterm:

```
db2inst1> db2 “connect to sample user  'uid=db2inst1'"
Enter current password for uid=db2inst1: <password>
```

This connection should be successful. Terminate your connection and connect with the full user DN and type the generated password when prompted:

```
db2inst1> db2 “connect to sample user
               'uid=db2inst1,ou=people,dc=example,dc=com'"
Enter current password for
  uid=db2inst1,ou=people,dc=example,dc=com: <password>
```

This connect statement should also be successful. Terminate your connection again and try another statement.

Connect with a part of the user DN and specify the password as parameter:

```
db2inst1> db2 “connect to sample user 'uid=db2inst1'
               using <password>”
```

Terminate your connection and exit the user shell.

As you can see, there are many ways to connect to your database and the good thing is that the authentication is absolutely transparent. You can use a connect statement without specifying a full user name and a password or just a part of it. Since we configured the

DB2-LDAP communication successfully, DB2 will always find the right LDAP user for your connection command.

6.6 Problem Determination (optional)

If you have problems with your LDAP configuration, you can debug the LDAP and DB2 behaviour. This chapter gives you a simple overview about the log files which are important for problem determination regarding DB2 and LDAP.

To debug the communication from the DB2 side, you can set the DEBUG parameter in the configuration file `IBMLDAPSecurity.ini` to `true`.

```
# vi /home/db2inst1/sqllib/cfg/IBMLDAPSecurity.ini
DEBUG = true
```

This leads to some extra information in the *db2diag.log* for LDAP related issues. Most of the additional information will be logged at *DIAGLEVEL 4* (INFO). So you can open the *db2diag.log* and check the DB2 related LDAP messages:

```
# vi /home/db2inst1/sqllib/db2dump/db2diag.log
```

On the other hand, if you have some general LDAP problems you can find helpful information in the system log */var/log/messages*. We have already used this in the steps before.

```
# vi /var/log/messages
```

The LDAP output depends on your *loglevel* which can be specified in the LDAP-server configuration file */etc/openldap/slapd.conf*. The default value for the log level is 0 which prevents LDAP related logging. You can choose between -1 (enable all logging), 0 (prevent logging) and more specific logging values like 1 (trace function calls), 8 (connection management), 32 (search filter processing) and so on. You can find a good overview and more detailed information about the log level in [5].

```
# vi /etc/openldap/slapd.conf
loglevel 0
```

If you have authorization problems with your LDAP user, you can use the following command to check if your users and groups really exist on your LDAP server:

```
# getent passwd
# getent group
```

7. Dedicated OpenLDAP and Database Server

For your first steps with LDAP and DB2 we recommend you to configure your LDAP and your DB2 server on the same host. However, if you decide to use LDAP and DB2 for a productive environment, you should use dedicated servers for both DB2 and LDAP. This ensures a better performance as well as easy extensibility, and prevents a possible bottleneck in your system landscape.

Clients can use the same LDAP user account to connect to your central database (see Figure 13 - Dedicated LDAP- and Database Server). In this chapter we will give you a short overview about the required adoptions regarding your system landscape.

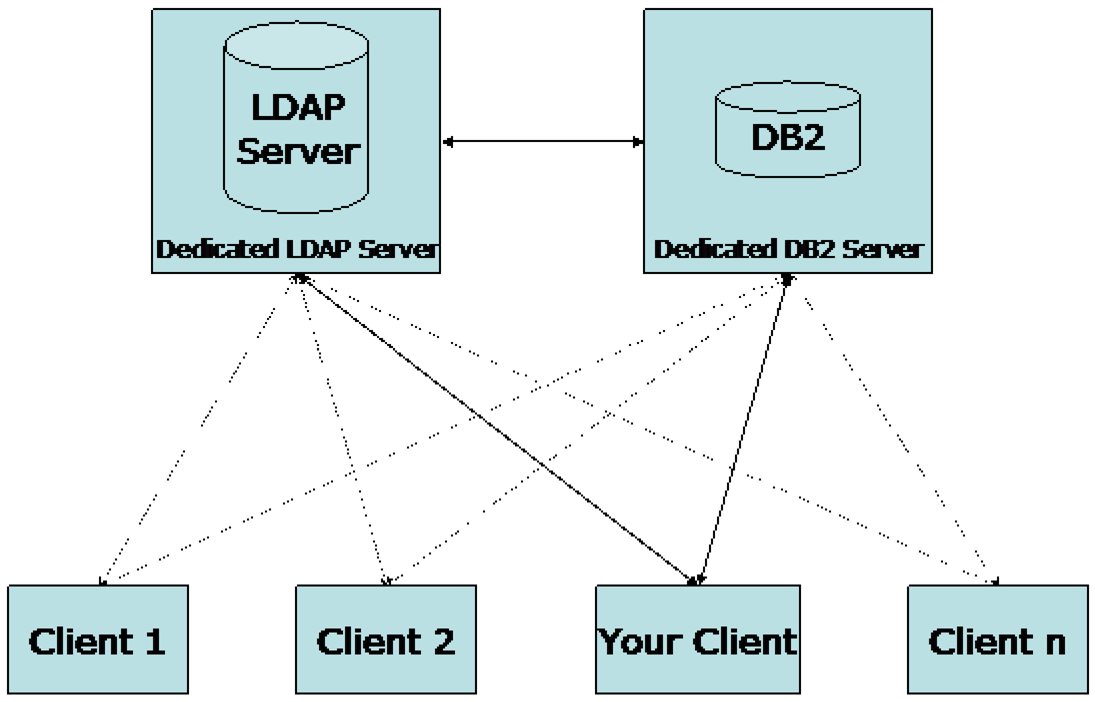


Figure 13 - Dedicated LDAP- and Database Server

7.1 Validate Remote OpenLDAP Environment

The LDAP server should be configured as described in section 5.1. This configuration is independent of your system environment. The same applies to the database server. The DB2 server can be configured as described in Chapter 6, except that you have to use a different value for the parameter *LDAP_HOST* in the configuration file *IBMLDAPSecurity.ini*. The value is the IP address or the hostname of your dedicated LDAP server.

So we will only need to focus on the clients. Most of the time, a client needs a connection to your remote LDAP and DB2 server. This requires that you configure the LDAP client as described in section 5.2 with the difference that your LDAP server is not local and you need to specify the IP address or the hostname again. On the other hand, we will need to configure the client to be able to access the remote database.

Before we do this, we want to validate the current LDAP configuration. Start with checking if the LDAP server is accessible via *ping:*

```
# ping <LDAP-server-ip>
64 bytes from <server-ip>: icmp_seq=1 ttl=64 time=0.624 ms
64 bytes from <server-ip>: icmp_seq=2 ttl=64 time=0.877 ms
64 bytes from <server-ip>: icmp_seq=3 ttl=64 time=0.945 ms
```

If your LDAP server is accessible, check if your LDAP client is correctly configured. Ensure this by using the following commands:

```
# getent passwd
```

Do you see your LDAP users?

```
# getent group
```

Do you see your LDAP groups?

```
# ldapsearch –x
```

Do you find any LDAP entries from the remote server?
If one of the above commands was not successful, debug and change your configuration.

7.2 Prepare Remote Database Access

We have two options to access the remote instance: we could create a local client instance and catalogue the remote database in this client instance, or we use a CLI-driver and get direct access to our remote database.

The additional client instance is not the smartest way, because you need two instances and have to catalogue the same database twice. By using a CLI-driver for your remote connection, you will have transparent access to your remote database without doing redundant work.

The good news is, that SAP changed is architecture with all products based on kernels greater than 7.00 to this new architecture, also called the change from the fat to the thin client. So if you want to use LDAP in a SAP system landscape, the SAP installation tool will automatically setup this structure for you.

8. Integrating SAP, DB2 and OpenLDAP

If you want to use LDAP authentication method for DB2 and SAP users in an SAP environment, we recommend integrating it into existing SAP environment instead of trying to reconfigure your LDAP users, before you want to set up a new SAP system. The authentication methods reconfiguration is much easier to handle **after** the installation.

8.1 Integrate LDAP Into Existing SAP/DB2 Environment

Configure your LDAP environment as described in chapter 5 - Configuration Of OpenLDAP. We recommend configuring a dedicated LDAP server (see 5.1) and at least one LDAP client (see 5.2) on the same machine. Thus, you could also use the OpenLDAP tools on the dedicated LDAP server to manage your LDAP directory.

Following is our proposal for the required SAP and DB2 users in a LDIF-like structure (db2_for_sap.ldif):

(replace the `<sid>` template with your SAP system's SID)

```
# base dn
dn: dc=example,dc=com
dc: example
o: example
objectClass: organization
objectClass: dcObject

# Hostname – For Uniqueness
# NOTE: Replace <host> with your hostname!
dn: dc=<host>,dc=example,dc=com
dc: <host>
o:  <host>
objectClass: organization
objectClass: dcObject

#
# DB2 SYSADM_GROUP
# NOTE: Replace <sid> with your system id!
dn:          cn=db<sid>adm,dc=<host>,dc=example,dc=com
cn:          db<sid>adm
objectClass: top
objectClass: posixGroup
gidNumber:   300
objectClass: groupOfNames
member:
   uid=db2<sid>,cn=db<sid>adm,dc=<host>,dc=example,dc=com
memberUid:   db2<sid>
```

```
#
# DB2 User: DB2 Database Administrator, db2<sid>
#
dn: uid=db2<sid>,cn=db<sid>adm,dc=<host>,dc=example,dc=com
cn:            db2<sid>
sn:            db2<sid>
uid:           db2<sid>
objectClass:   top
objectClass:   inetOrgPerson
objectClass:   posixAccount
uidNumber:     300
gidNumber:     300
loginShell:    /bin/csh
homeDirectory: /db2/db2<sid>

#
# SAP Group: db<sid>mnt
#
dn:          cn=db<sid>mnt,dc=<host>,dc=example,dc=com
cn:          db<sid>mnt
objectClass: top
objectClass: posixGroup
gidNumber:   301
objectClass: groupOfNames
member:
    uid=sap<sid>,cn=db<sid>mnt,dc=<host>,dc=example,dc=com
memberUid:   sap<sid>

#
# SAP User: ABAP Connect User, sap<sid>
#
dn: uid=sap<sid>,cn=db<sid>mnt,dc=<host>,dc=example,dc=com
cn:            sap<sid>
sn:            sap<sid>
uid:           sap<sid>
objectClass:   top
objectClass:   inetOrgPerson
objectClass:   posixAccount
uidNumber:     301
gidNumber:     301
loginShell:    /bin/csh
homeDirectory: /home/sap<sid>

#
# SAP Group: db<sid>ctl
#
dn:          cn=db<sid>ctl,dc=<host>,dc=example,dc=com
cn:          db<sid>ctl
objectClass: top
objectClass: posixGroup
gidNumber:   302
objectClass: groupOfNames
member: uid=<sid>adm,cn=sapsys,dc=<host>,dc=example,dc=com
memberUid:   <sid>adm
```

```
#
# SAP Group: sapsys
#
dn:          cn=sapsys,dc=<host>,dc=example,dc=com
cn:          sapsys
objectClass: top
objectClass: posixGroup
gidNumber:   303
objectClass: groupOfNames
member: uid=<sid>adm,cn=sapsys,dc=<host>,dc=example,dc=com
memberUid:   <sid>adm

#
# SAP User: System Administrator, <sid>adm
#
dn:     uid=<sid>adm,cn=sapsys,dc=<host>,dc=example,dc=com
cn:            <sid>adm
sn:            <sid>adm
uid:           <sid>adm
objectClass:   top
objectClass:   inetOrgPerson
objectClass:   posixAccount
uidNumber:     302
gidNumber:     303
loginShell:    /bin/csh
homeDirectory: /home/<sid>adm
```

Figure 14 - LDIF for DB2 and SAP Users

You can use the following two scripts to easily add and remove LDAP objects. Use the script template above to customize it to your needs and then add the DB2 and SAP users to LDAP server.

If you have tried the sample steps earlier in this Whitepaper and therefore created some users on your test LDAP server, you can recursively delete **ALL users** registered in LDAP by executing the following command:

```
ldapdelete -v
           -x
           -D "cn=Manager,dc=example,dc=com"
           -W
           -r "dc=example,dc=com"
```

Figure 14 - del_db2_for_sap.sh

After that, add the pre-defined LDAP users from the “db2_for_sap.ldif” script with the following command:

```
  ldapadd -x -D "cn=Manager,dc=example,dc=com" –W -f
   /<path>/db2_for_sap.ldif
```

Figure 15 – add LDAP users from the LDIF template

Note that you have to write the whole command into one line!

Before you continue with the chapter, stop your SAP system and the underlying database. Otherwise you may run into database resource problems, caused by your local database processes:

```
  # su - <sid>adm
  <sid>adm# stopsap
  # exit
```

8.2 Database Administrator

Ensure that you have created two database administrator UserID’s: one as a local user and one as LDAP user, both with the same username:

Note that you can easily distinguish between the LDAP users and the local ones, because all LDAP users have the ‘*’ symbol as the second element in the “getent” command output.

```
   # getent passwd | grep db2<sid>
   db2<sid>:x:1006:400:Database Admin:/db2/db2<sid>:/bin/csh
   db2<sid>:*:300:300:/db2/db2<sid>:/bin/csh
```

Similarly, validate the existence of two *db<sid>adm* groups:

```
   # getent group | grep db<sid>adm
   db<sid>adm:!:400:
   db<sid>adm:*:300:db2<sid>
```

Customize the configuration file *IBMLDAPSecurity.ini* as described in section 6.4. This results in a file similar to the following:

```
 ;-----------------------------------------------------------
 ; SERVER RELATED VALUES
 ;-----------------------------------------------------------
 LDAP_HOST = <remote-ip>

 ;-----------------------------------------------------------
 ; USER RELATED VALUES
 ;-----------------------------------------------------------
 USER_OBJECTCLASS = posixAccount
```

```
USER_BASEDN = dc=example,dc=com

USERID_ATTRIBUTE = uid

AUTHID_ATTRIBUTE = uid

;-----------------------------------------------------------
; GROUP RELATED VALUES
;-----------------------------------------------------------

GROUP_OBJECTCLASS = groupOfNames

GROUP_BASEDN = dc=example,dc=com

GROUPNAME_ATTRIBUTE = cn

GROUP_LOOKUP_METHOD = SEARCH_BY_DN

GROUP_LOOKUP_ATTRIBUTE = member
```

Update the database configuration manager (DBM) to use the LDAP plug-ins as your local database administrator:

```
 # su – db2<sid>
```

Note, if you execute the *id* command now, you should see that your *uid* is the same *id* as your **local** database administrator. In our case it is 1006 instead of 300.

If you see the userID “300”, this indicates that the system authentication method is currently set to LDAP instead of local. You should temporarily switch the authentication back to local in YaST.

This is important, because right now the local database administrator is the only one allowed to update the DBM configuration.

Update the DBM configuration for usage with LDAP plugins:

```
 db2<sid># db2 update dbm cfg using
                SRVCON_PW_PLUGIN IBMLDAPauthserver
                GROUP_PLUGIN     IBMLDAPgroups
 db2<sid># exit
```

Before you continue, rename the local database user *db2<sid>* in */etc/passwd* and his group *db<sid>adm* in */etc/group*. This ensures unique names and helps you to understand who you are (local or LDAP user).

Now you can change the ownership of the database directories to the new LDAP user and his group:

```
 # chown -R db2<sid>:db<sid>adm /db2/db2<sid>/
 # chown -R db2<sid>:db<sid>adm /db2/<SID>/
```

After you have renamed the local database administrator and his group, you can switch to your LDAP user. You should check with the *id* command, whether this is really the LDAP user (in our case *uid*=300):

```
# su – db2<sid>
db2<sid># id
uid=300(db2<sid>) gid=300(db<sid>adm) groups=300(db<sid>adm)
```

Try now if your LDAP user has the privileges to start the database instance and issue a test connect statement to your database:

```
db2<sid># db2start
SQL1063N DB2START processing was successful.
db2<sid># db2 connect to <db>
    Database Connection Information
 Database server        = DB2/LINUXX8664 9.5.1
 SQL authorization ID   = DB2<SID>
 Local database alias   = <SID>
```

As you could see, your LDAP database administrator is correctly configured.

8.3 Database Connect User

After you configured your database to use LDAP, all users accessing your database have to use LDAP too. In this chapter, we will demonstrate the required changes for the ABAP Connect User *sap<sid>*. Similar changes may be required for the JAVA Connect User *sap<sid>db* if you have a Java stack in your SAP system:

Ensure again, if you have a local and a LDAP database connect user (and his group) on your machine configured and rename the local connect user and his group:

```
# getent passwd | grep sap<sid>
sap<sid>:x:1007:1006:ABAP Database Connect
User:/home/sap<sid>:/bin/csh
sap<sid>:*:301:301:ABAP Database Connect
User:/home/sap<sid>:/bin/csh
# userdel sap<sid>
# getent group | grep
db<sid>mnt:!:1006:
db<sid>mnt:*:301:sap<sid>
# groupdel db<sid>mnt
```

Change the file system permissions to the new LDAP user id and his group id:

```
# chown -R sap<sid>:db<sid>mnt /home/sap<sid>
```

Set the LDAP password for this user to the password of the local database connect user. We need to have the same password, because SAP will validate the password with the existing one in /sapmnt/<SAPSID>/global/dscdb6.conf.

```
db2adm# ldappasswd –x
    -D “cn=Manager,dc=example,dc=com”
    -W
    "uid=sap<sid>,cn=db<sid>mnt,dc=<host>,dc=example,dc=com"
    –S
 New password:
 Re-enter new password:
 Result: Success (0)
```

Validate your database connection with the credentials of your ABAP Connect User:

```
# su – db2<sid>
db2<sid># db2 connect to <db> user sap<sid> using <passwd>
   Database Connection Information
 Database server        = DB2/LINUXX8664 9.5.1
 SQL authorization ID   = SAP<SID>
 Local database alias   = <DB>
```

8.4 SAP System Administrator

The last user is the system administrator of the SAP system. Ensure the existence of the LDAP and the local user accounts for the SAP system administrator on your machine:

```
# getent passwd | grep <sid>adm
<sid>adm:x:1005:1004:SAP System
Administrator:/home/<sid>adm:/bin/csh
<sid>adm:*:303:303:SAP System Administrator:
   /home/<sid>adm:/bin/csh
```

Note that this user is related to two groups. The primary group is *sapsys* and the secondary *db<sid>ctl*. So check if these groups exist on your local system and in your LDAP directory:

```
# getent group | grep sapsys
sapsys:!:1004:
sapsys:*:303:<sid>adm
# getent group | grep db<sid>ctl
db<sid>ctl:!:1005:<sid>adm
db<sid>ctl:*:302:<sid>adm
```

Rename the local user and his both local groups.

Change the file system permissions to the new LDAP user id and group id:

```
# chown -R <sid>adm:sapsys /home/<sid>adm
# chown -R <sid>adm:sapsys /usr/sap/<SID>
# chown -R <sid>adm:sapsys /sapmnt/<SID>
```

Note that we have also got some other directories in */usr/sap* which are owned by root and are related to the group *sapsys* (e.g. */usr/sap/trans*). You should also change this group from the local one to the new LDAP *sapsys* group.

The exact amount of subdirectories in */usr/sap/* depends on your SAP product configuration. Execute the "`ls -la /usr/sap/`" command to see which directories are owned by the local userID of the *sapsys* user, and then perform the "`chown`" command to change ownership to the appropriate LDAP user.

Validate the database credentials of your SAP system administrator via R3trans:

```
# su – <sid>adm
<sid>adm> R3trans –x
<sid>adm> vi trans.log
```

If you find a successful database connection like "*Connected to DB2 server type 'DB2/LINUXX8664'*" at the bottom of the file, you have successfully completed your LDAP configuration.

You just need to start your SAP system now to test the modification on the application level. So log into your SAP system and work as usual.

We recommend starting the transaction DBACOCKPIT to validate whether the application server works properly with the database server.

9. List of Abbreviations

LDAP	Lightweight Directory Access Protocol
DN	Distinguished Name
PAM	Pluggable Authentication Module
POSIX	Portable Operating System Interface
NSS	Network Security Services
LDIF	LDAP Data Interchange Format
CLI	Client Library Interface

10. Table of Figures

11. List of Literature

[1] **OpenLDAP Administrator's Guide**

http://www.openldap.org/doc/

[2] **Understanding LDAP – Design and Implementation, June 2004,**

http://www.redbooks.ibm.com/abstracts/sg244986.html

[3] **RFC1558 - Backus-Naur-Form (BNF)**

http://rfc.net/rfc1558.html

[4] **RFC2222 - Simple Authentication and Security Layer (SASL)**

http://rfc.net/rfc2222.html

[5] **Open Source Guide about LDAP**

http://www.zytrax.com/books/ldap/ch6/#loglevel