# Writing LDAP Entries to an LDIF File - *(Using ldapwrite and ldapsearch)*

*by Jeff Hunter, Sr. Database Administrator*

---

## Contents



---

## Overview

There are two easy ways in which to write LDAP entries (application data) to an LDIF flat file. Those two ways are the command-line utilities **`ldifwrite`** or **`ldapsearch`**.

### Using `ldifwrite`

`ldifwrite` is intended for use only on OID "application data", not the schema or operational attributes and values (e.g., cn=catalogs, cn=changelog, etc.). Keep in mind that in order to use the command-line utility, you will need to know and use the `ODS` database username and password. The syntax for the `ldifwrite` command is:

```
ldifwrite -c <db_connect_string> -b <base DN> -f <filename>
```

### Exporting Application Data and Values

You can type the following commands to export OID application data using `ldapwrite`:

```
# ldifwrite -c OIDDB_ALEX -b "o=airius.com" -f airius.ldif
This tool can only be executed if you know database user password for OiD
Enter OiD Password :: ods
```

### Exporting Schema or Operational Attributes and Values

As previously mentioned, you cannot use `ldifwrite` to write out schema or operational data. In order to extract schema and operational data, you will need to use `ldapsearch` (described below). Trying to perform a search of this type, will result in the following error:

```
# ldifwrite -c OIDDB_ALEX -b "cn=subschemaSubentry" -f alex_oid_schema.ldif
This tool can only be executed if you know database user password for OiD
Enter OiD Password :: ods
Base DN cn=subschemasubentry not found.
```

### Using `ldapsearch`

The syntax for the `ldifsearch` command, while writing out to an LDIF file is:

```
ldapsearch -L -D "cn=orcladmin" -w "welcome" -h <host> -p <port> -b <base
DN> -s <Search Scope (base | one | sub)> "<Search Filter>" >
output_file.ldif
```

### Exporting Application Data and Values

You can type the following commands to export OID application data using `ldapsearch`:

```
# ldapsearch -L -D "cn=orcladmin" -w "welcome" -h alex -p 389 -b
"o=airius.com" -s sub "objectclass=*" > airius.ldif
```

### Exporting Schema or Operational Attributes and Values

As previously mentioned, you cannot use `ldifwrite` to write out schema or operational data. You can only use to perform this type of operation. To export the schema or operational attributes and values, you need to use ldapsearch with the `-L` option as follows:

```
# ldapsearch -L -D "cn=orcladmin" -w "welcome" -h alex -p 389 -b
"cn=subschemaSubentry" -s base "objectclass=*" > alex_oid_schema.ldif
```

---

## Installing Oracle Internet Directory

*by Jeff Hunter, Sr. Database Administrator*

### Contents



### Overview

**OID Installation Overview**

The following section deals with installation and setup issues while installing Oracle Internet Directory (OID) Version 9.2.0. This version of OID comes packaged on the Oracle Enterprise Server RDBMS 9.2.0 CDs.

The DBA will need to perform two separate installations:

1. The Oracle 9.2.0 Database Product Set
2. Oracle Internet Directory Server (9.2.0) within the same ORACLE_HOME as the Oracle 9.2.0 Database Product Set (above).

## OID Installation Overview

- OID 9.2.0 is bundled with Oracle9i Enterprise Edition on the latest cd pack.
- Oracle Internet Directory is now licensed as part of Oracle9*i* Application Server
- Oracle Internet Directory is also included in the Oracle9*i* Data Server Media Pack.
- As part of the Database bundle, customers receive a Restricted User license to support other components of the database.
- OID 9.2.0 or 3.0.1 is not currently available on the Oracle9i Application Server Release 1 media pack.
- Customers requiring 9.2.0 of OID must install it from the Oracle9*i* Database Release 1 media.
- OID 9.2.0 runs only on Oracle9*i* Databases, and is neither certified, nor supported with Oracle8*i* or any Oracle9*i*AS Release 1 components.

**NOTE:** The version of Oracle Internet Directory included with Oracle9*i* Release 1 (9.0.1.0.0) is *Oracle Internet Directory (OID) 3.0.1*. The installation procedure for OID 3.0.1 is exactly the same as that for OID 9.2.0.

## UTF-8 Considerations

The Oracle directory server and database tools are no long restricted to run on a UTF8 database.

### Using Non-UTF-8 Databases

You can run the Oracle directory server and database tools on a non-UTF-8 database, but be sure that the client and database character sets are the same. Otherwise, you can loose data during **`ldapadd`**, **`ldapdelete`**, **`ldapmodify`**, or **`ldapmodifydn`** operations. For example, suppose that you perform an **`ldapadd`** operation using a multibyte character set on an underlying database that uses only a single-byte characters. You will lose data because not all of the bytes you enter will be accepted by the database.

## Training and Demos

Take a look in the following directory:
`$ORACLE_HOME/ldap/demo/samples/training`

## Installing OID

### Install Oracle Enterprise Edition Database Product Set

Before installing the Oracle Internet Directory (OID), the DBA will need to perform an Oracle9*i* Release 2 (9.2.0.1.0) installation.

The OID product should be installed in the same ORACLE_HOME created in the Oracle9*i* installation *(above)*.

I typically name the ORACLE_HOME "`9.2.0`" or "`OIDMGR`". This will typically be the only product running against the database.

After installing the Oracle9*i* Release 2 product set and applying any required patches, create the database that will be used by OID. I typically name the database: `ORACLE_SID=OIDDB`.

**NOTE:** It is HIGHLY recommended for production instances of OID, that it be contained on a server dedicated to only OID. No other product or application should coexist on this machine.

After installing the Oracle9*i* Release 2 product set, applying any database patches and creating the database, the DBA should be ready to install the OID product and schema.

**NOTE !!!!!!!! :** DO NOT change the password for the SYSTEM account before installing the OID product. The password needs to be set to MANAGER in order for the OID Configuration Assistant to install the base schema.

## Install Oracle Internet Directory

**NOTE:** Before installing Oracle Internet Directory, ensure that you have followed the instructions in the above section: *Install Oracle Enterprise Edition Database Product Set*. Also make sure that the SYSTEM password is set to MANAGER.

### *Installation Phase*

1. For installing OID 9.2.0, ensure that the ORACLE_HOME is set the Oracle9*i* Release 2 installation performed in the above section.
2. I had problems when using the Oracle 9.2.0.3.0 patchset. I first performed the Oracle9*i* Release 2 installation, then installed the 9.2.0.3.0 patchset, and finally installed the Oracle Internet Directory. During the OID Configuration process, it failed after the second screen, indicating that it could not connect to the OID LDAP Server. I removed all of the installed products, installed Oracle9*i* Release 2, did NOT installed the 9.2.0.3.0 patchset, and then successfully installed OID.
3. If you plan on using the default ports for LDAP RFC standards, (LDAP Port = 389, and LDAP SSL Port = 636), you may need to check that no application is using those ports AND that any entries in the `/etc/services` file defined for those ports (both TCP and UDP) are removed from the file before the OID Configuration process is run.
4. Run the "**`runInstaller`**" shell script from the Oracle9*i* Release 2 Enterprise Edition Product Set.
5. On the "*Welcome*" screen, hit "*Next*".
6. On the "*File Locations*" screen, do not change anything. Hit "*Next*".
7. On the "*Available Products*" screen, choose the "*Oracle9i Management and Integration 9.2.0.1.0*" radio button and hit "*Next*".
8. On the "*Installation Types*" screen, select "*Oracle Internet Directory*" and hit "*Next*".
9. Since you already created an Oracle 9*i* database using this ORACLE_HOME, you will be prompted by a screen called "*Using an existing instance*". Choose "*Yes*" and hit "*Next*".
10. In the "*Database Identification*" screen, type the name of the database instance you created. (i.e. OIDDB).
11. In the "*OID Database File Location*" screen, make sure that the directory is set to: "`/u10/app/oradata/<SID_NAME>/oradata`". Hit "Next".
12. On the "*Summary*" screen, hit "*Install*" to start the installation and linking phase.

***Setting Values***

The following values are automatically set during installation:

| Setting | Value |
|---|---|
| Use of an Encrypted Password | Yes |
| Encryption Schema | MD4 |
| Approximate number of directory entries to be stored inOracle Internet Directory | Under 10,000 entries |
| Password of the Administrator Distinguished Name | welcome |

***Running root.sh***

The Installer creates the root.sh script in the Oracle home directory and prompts you to run the script when it finishes installing Oracle products. The **`root.sh`** script sets the necessary file permissions for Oracle products and performs other root-related configuration activities. Log in as the root user and run the script. To run the **`root.sh`** script enter the following commands:

```
# cd $ORACLE_HOME
# ./root.sh

Entering Oracle Internet Directory Root Installation Section

OiD Server Installation
Checking LDAP binary file protections
Setting oidmon file protections
Setting oidldapd file protections
Setting oidrepld file protections
Setting oidpasswd file protections
Setting oidstats.sh file protections
Setting oidpwdr file protections
Setting odisrv file permissions
Leaving Oracle Internet Directory Root Installation Section
```

If you install Oracle9i Real Application Clusters, you must run the **`root.sh`** script on every node in the cluster.

When the **`root.sh`** script runs successfully, return to the Oracle Universal Installer. Click `OK` in the *Alert window*.

***Configuration Assistant***

The "Configuration Tools" assistant appears at the end of the installation and automatically starts the OID Configuration Assistant. The OID Configuration Assistant is a series of screens that significantly reduces the complexity of configuring the OID.

**Screen 1**

The first screen simply provides the login credentials. The values are already included and in many cases, you can simply hit the *Next* button. After hitting the *Next* button, another dialog box is presented that states "*Please wait...*". This process takes several minutes to complete.

**Screen 2**

The second screen allows you to enter the OID Server Details. By default, the *OID Port* defaults to **`389`**, while the default *OID SSL Port* is set to **`636`**. If the OID Configuration process fails to display the default ports, this means that the ports were not available for use. If this is the case, the OID Configuration Assistant will get a free port in the range - 4031 to

4039. *(The installer will typicall choose 4032 for the non-SSL port and 4031 for the SSL port. In most cases, there were entries in the `/etc/services` file for ports 389 and 636 (both TCP and UDP.)*

**FONT:** If you want to use the LDAP RFC standard ports (389 / 636), you will need to exit from the OID Configuration Assistant and Oracle Installer. I was able to exit from the Oracle Universal Installer, remove the entries in the `/etc/services` file, and make modifications to the script: `$ORACLE_HOME/ldap/postcfg/postcfg`. After you removed the entries in the `/etc/services` file for the 389 and 636 ports, you can safely modify the last line of the script to use the default ports from 4032(non-SSL) and 4031(SSL) to 389(non-SSL) and 636(SSL). You would then re-run the script:

```
% $ORACLE_HOME/ldap/postcfg/postcfg
```

After clicking on the *Next* button, you will once again, be presented with the "*Please wait...*" dialog box. This process will also take several minutes to complete.

**Screen 3**

If everything goes well, you will be presented with a third and final screen indicating that the OID installation was successfully completed. Simply click the "*Finish*" button and exit from the Oracle Universal Installer.

---

**NOTE:** Linux Users, after the installation of the Oracle Internet Directory, run the **`postcfg`** script from the command line to run the OID post-installation configuration steps:

```
export JAVA_HOME=/u01/app/oracle/jre/1.1.8
cd $ORACLE_HOME/ldap/postcfg
postcfg
```

Start the oidmon process at the command prompt.

Start an oidldapd process using the **`oidctl`** utility at the command prompt.

Now import the data. The following imports the data into a machine named *cartman*:

```
ldapmodify -c -a -v -h cartman -D "cn=orcladmin" -w "welcome" -f
oidbaseacl.ldif
ldapmodify -c -a -v -h cartman -D "cn=orcladmin" -w "welcome" -f
oidbase.ldif
ldapmodify -c -a -v -h cartman -D "cn=orcladmin" -w "welcome" -f
oidnet.ldif
ldapmodify -c -a -v -h cartman -D "cn=orcladmin" -w "welcome" -f
oidrdbms.ldif
```

**NOTE:** The LDAP schema loading is done automatically at the end of the installation. If this step does not go through, then the following `ldif` files should be loaded into the directory IN THE ORDER LISTED, using **`ldapmodify`** at the command line:

| File Name | Description |
|---|---|
| `$ORACLE_HOME/ldap/admin/oidbaseacl.ldif` | This implements the default security policy. |
| `$ORACLE_HOME/ldap/admin/oidbase.ldif` | This loads the common schema required by all Oracle LDAP enabled products. |
| `$ORACLE_HOME/ldap/admin/oidnet.ldif` | This loads the schema required for LDAP support in Net8. |
| `$ORACLE_HOME/ldap/admin/oidrdbms.ldif` | This loads the schema required for Oracle8i RDBMS to use Oracle Internet Directory. |

**OID Configuration Assistant - How to create a new OID schema manually**

The purpose of this section is to describe how to create the database components required by Oracle Internet Directory, and how to create the Oracle directory schema and its' extensions in the directory database without installing the product again.

**NOTE:** This operation can only be performed against a properly created database. One might need to use this procedure in order to test OID using a new database, or after a failed install. However, it's important to remember, that this procedure can be used ONLY if oracle binaries/executables are properly installed. This procedure must be performed completely without leaving any steps out. To avoid problems with install, it is suggested that the Oracle Internet Directory Installation Guide and the notes referenced at the end of this article are reviewed.

***OID Configuration Assistant***

At the time of installation, something called OID Configuration Assistant is executed. This is not a real utility, so there's no binary/executable to be started to run this tool again. It's possible that in some future release, this utility will be included.

The following describes how to do this manually. It is assumed that the database has been created either during the install or by using the Oracle Database Configuration Assistant.

1. Stop your LDAP server(s), replication server and oidmon if they are running.
2. In case the purpose of this exercise is to restore the OID database to a clean state, as it was after install, all the tablespaces created by the script `newldapcre.sql` must be dropped. See `newldapcre.sql` for a complete list of tablespaces, and use the following command to drop them, one at the time.

   ```
   SQL> DROP TABLESPACE <TABLESPACE_NAME> INCLUDING
   CONTENTS;
   ```

   If this is not done, all statements in `newldapcre.sql` will fail. Then drop OID database users with following commands:

   ```
   SQL> DROP USER ODS CASCADE;
   SQL> DROP USER ODSCOMMON CASCADE;
   ```

3. With sqlplus, run the following sql script when logged in as system.

   On Unix:

   ```
   SQL> $ORACLE_HOME/ldap/admin/newldap.sql
   ```

   On Windows:

   ```
   SQL> %ORACLE_HOME%\ldap\admin\newldap.sql
   ```

This script will call other scripts which will then create all the tablespaces needed by OID, all the users, and all the database objects for them. If you want to store datafiles in a directory other than

```
$ORACLE_HOME/dbs/oradata/<SID> (Unix) or
%ORACLE_HOME%\dbs\oradata\<SID> (Windows)
```

modify `newldapcre.sql` accordingly before running `newldap.sql`.

4. If the any of the following patchsets 2.1.1.1, 2.1.1.2 or 2.1.1.3 have been applied do the following steps as well (Solaris only).
    1. If using 2.1.1.1 binaries:

        Run The Patch Configuration Assistant.

        **`$ORACLE_HOME/ldap/install/schema2111.sh`**

        Before you run the script make sure that the ORACLE environment is set and that the Oracle Internet Directory server is not running.

        Database and Listener must be up and running. The usage for this script is as follows:

        ```
        schema2111.sh -odspwd <ODS userpassword>
                      -sudn <Oracle Internet Directory
        superuser DN>
                      -supwd <Oracle Internet Directory
        superuser password>
        ```

    2. If using 2.1.1.2 binaries

        Do everything listed in step 4.1, as 2.1.1.2 can only be applied on top of 2.1.1.1.

        Login as ODS with sqlplus, and execute thw following sql script:

        `SQL>` **`$ORACLE_HOME/ldap/admin/ldapu2112.sql`**

    3. If using 2.1.1.3 binaries

        Do everything listed in step 4.1. 2.1.1.3 can be applied on top of 2.1.1.1 or 2.1.1.2, so there's no need to do step 4.2.

5. Start oidmon and LDAP server.
6. Use ldapmodify to load the Oracle schema in the directory.

    The following ldif files need to be loaded, in the same order they are listed below. All files can be found in directory:

    `$ORACLE_HOME/ldap/admin (Unix) or`

```
%ORACLE_HOME%\ldap\admin (Windows).

   oidbaseacl.ldif   -> this implements the default
security policy.
   oidbase.ldif      -> this loads the common schema
required by all
                        Oracle LDAP enabled products.
   oidnet.ldif       -> this loads the schema required
for LDAP support
                        in Net8.
   oidrdbms.ldif     -> this loads the schema required
for Oracle8i
                        RDBMS to use Oracle Internet
Directory.
```

e.g.

```
   ldapmodify -h <host> -p 389 -D "cn=username" -w
"password" -c -v -f oidbaseacl.ldif
   ldapmodify -h <host> -p 389 -D "cn=username" -w
"password" -c -v -a -f oidbase.ldif
   ldapmodify -h <host> -p 389 -D "cn=username" -w
"password" -c -v -a -f oidnet.ldif
   ldapmodify -h <host> -p 389 -D "cn=username" -w
"password" -c -v -a -f oidrdbms.ldif
```

If using OID 9.2.0 or 3.0.1, last three ldif files can be loaded with Net Configuration Assistant by doing the following:

```
- start Net Configuration Assistant (NetCA)
- select "Directory Usage Configuration"
- select "Create or upgrade the Oracle Schema (Advanced)"
- select "Oracle Internet Directory" as a directory type
- provide hostname, port number, and SSL port number
- provide user credentials to login to the directory.
User DN should be
  cn=orcladmin and password welcome. Note that using user
DN without
  "cn=" will cause Authentication Error.
```

## Starting and Stopping OID

### The OID Monitor Process

The OID Monitor must be running to process commands to start and stop the server.

OID Monitor is a component that initiates, monitors, and terminates the Oracle directory server processes.

It also controls the replication server if one is installed, and the Oracle directory integration server.

### The Server Instances

The OID Control Utility, "**`oidctl`**" is a command-line tool for issuing run-server and stop-server commands.

The commands are interpreted and executed by the **OID Monitor** process.

**Scripts used to Start/Stop the directory services**

I created two scripts that can be used to start and stop the Oracle Internet Directory Server:

- **start_OID.sh**

  Starts the OID Monitor Process as well as any "oidldap" Server Instances. This performs the following tasks:

  - Switches the Oracle Environment to the ORACLE_HOME of the OID installation.
  - Starts the OID Monitor Process:

    ```
    % oidmon connect=OIDDB start
    ```

  - Starts any Server Instances using the OID Control Utility:

    ```
    % oidctl connect=OIDDB server=oidldapd instance=1 start
    ```

- **stop_OID.sh**

  Stops the OID Monitor Process as well as any "oidldap" Server Instances. This performs the following tasks:

  - Switches the Oracle Environment to the ORACLE_HOME of the OID installation.
  - Stops any Server Instances using the OID Control Utility:

    ```
    % oidctl connect=OIDDB server=oidldapd instance=1  stop
    ```

  - Stops the OID Monitor Process:

    ```
    % oidmon connect=OIDDB stop
    ```

## Using Oracle Internet Directory Manager

### Overview

Oracle Directory Manager is a Java-based tool for administering Oracle Internet Directory. This section describes some of its basic features. More specific instructions are found in sections throughout this book that explain how to perform various tasks.

### Starting Oracle Internet Directory Manager

Before you can launch Oracle Directory Manager, you must have a directory "directory server instance" running.

To start Oracle Directory Manager, follow the instructions for your operating system:

**Windows NT or Windows 95**

From the Start menu, click:

```
  Programs > ORACLE_HOME > Oracle Internet Directory > Oracle Directory
Manager
```

**Sun Solaris**

If you have not set the path, then navigate to ORACLE_HOME/bin.

Type at the system prompt:

```
% oidadmin
```

**NOTE:** DO NOT try to launch *Oracle Internet Directory Manager* from the OEM console. This is a completely different version of the OID Manager and is not functional. Oracle expects to the this fixed in version 10*g*.

**Using Oracle Directory Manager**

The first time you start *Oracle Directory Manager*, an alert tells you that you must connect to a server. Click OK. The *Directory Server Connection* dialog box appears.

**Connecting to a directory server**

To connect to a directory server:

1. In the *Directory Server Connection* dialog box, type the name and port number of an available server.

   The default port is 389. You can change the port if you wish. However, if you have an Oracle directory server running on a port that is not the default, then be sure that any clients that use that server are informed of the correct port.

   Click OK. The *Oracle Directory Manager Connect* dialog box appears.

2. In each field of the Credentials tab page, type the information specific to this server instance as described in the next table.
   - **User**:

     The first time you log in, do so either as the super user or anonymously. If you intend to configure SSL features during this session, login as the **super user**.

     If you are logging in as the super user, in the User box, type:

     ```
     cn=orcladmin
     ```

     If you are logging in anonymously, leave the User box empty.

     If you have already set up the user's entry by using LDAP command-line tools, you can enter that user's entry in one of two ways:

     - Browse and select that entry by using the button to the right of the User field
     - Type the distinguished name (DN) for that user's entry by using the correct format, for example:

       ```
       cn=Jeff Hunter,ou=ENG,dc=idevelopment,dc=info
       ```

   - **Password**:

     If you are logging in as the super user and you specified a password for the super user during installation, in the Password box, type the password you specified.

Otherwise, type the default password, namely:
"**`welcome`**".

After you are logged into Oracle Directory Manager and have connected to a directory server, you should change this password to protect the directory.

If you are logging in anonymously, leave the Password box empty. If you want to login as a specific directory user, enter the corresponding password.

- **Server**:

  From the Server list, select the host containing the directory server to which you want to connect.

  If you are already connected to a directory server, and you want to connect to one on a different host:

  1. Click the button to the right of the Server field. The Select Directory Servers dialog box displays a list of available servers.
  2. Select a server.
  3. Click OK.

  To add a directory server to the list:

  4. In the Select Directory Servers dialog box, click Add. The Directory Server Connection dialog box appears.
  5. In the Server field, type the name of the directory server you want to add.
  6. In the Port field, type the port number for the server you want to add.
  7. Click OK. The added directory appears in the list in the Select Directory Server dialog box.

  To modify a directory server on the list:

  8. Select the directory server you want to modify.
  9. Click Edit. The Directory Server Connection dialog box appears.
  10. Modify the Server and Port fields, then click OK. The modifications for that server appear in the list in the Select Directory Server dialog box.

- **Port**:

  The default port (389) appears in this field. If there is more than one directory server instance on the same host, each directory server instance has a different port, and that port number appears in this field when you select the directory server instance.

  To change this port number:

0. Click the button to the right of the Server field.
1. In the Select Directory Server dialog box, select the directory server.
2. Click Edit. The Directory Server Connection dialog box appears.
3. In the Directory Server Connection dialog box, in the Port field, enter the new port number, then click Ok.

- **SSL Enabled**:

  Selecting this check box causes all commands you issue by using Oracle Directory Manager to be sent over Secure Sockets Layer (SSL). You can connect to a directory server either with or without SSL. If you connect by using SSL, then Oracle Directory Manager becomes an SSL client.

  You can connect in this way if both of the following two conditions are met:

  1.) The server to which you are connecting uses SSL. If that server does not use SSL, and you select this check box, then authentication will fail.

  2.) You have already created a wallet containing a certificate and a list of trusted certificates.

## De-installing OID

1. In order to properly De-install OiD properly (versions 2.x.x, 3.x.x, or 9.2.0), it must be done using the following steps:
2. Stop your oidldapd processes with the oidctl command line interface
3. Stop your oidmon processes with the oidmon command line interface
4. Drop the OiD database Schema in the database. See the following example Oracle SQL script that removes the OID schema. (Note: Make sure you have a good cold backup in case you want to reinstall.)
5. Use the Oracle Universal Installer to completely remove OiD.

```
*************** SCRIPT TO DROP OiD SCHEMA OBJECTS: ****************

rem Drop Tablespaces for ODS Schema
drop user ODSCOMMON cascade;
drop user ODS cascade;
drop tablespace olts_attrstore including contents and datafiles;
drop tablespace olts_ct_dn including contents and datafiles;
drop tablespace olts_ct_cn including contents and datafiles;
drop tablespace olts_ct_objcl including contents and datafiles;
drop tablespace olts_ct_store including contents and datafiles;
drop tablespace OLTS_TEMP including contents and datafiles;
drop tablespace olts_default including contents and datafiles;
drop tablespace olts_ind_attrstore including contents and datafiles;
drop tablespace olts_ind_ct_dn including contents and datafiles;
drop tablespace olts_ind_ct_cn including contents and datafiles;
drop tablespace olts_ind_ct_objcl including contents and datafiles;
drop tablespace olts_ind_ct_store including contents and datafiles;
```

## Troubleshooting

**Troubleshooting Start / Stop of Oracle Internet Directory**

***Overview***

The purpose of this article is to describe how the Oracle Internet Directory start / stop mechanism works, and with that information, help to troubleshoot possible problems with start and stop of the OID server.

This article is for everyone who needs to start / stop Oracle Internet Directory servers, oidldapd and oidrepld and concentrates mainly on oidldapd, but the same theory applies to oidrepld as well.

***Tools and Process Architecture***

Before we can successfully solve problems related to start / stop of Oracle Internet Directory servers, we need to know what is the purpose of every tool involved, and how those tools work together. Also, in order to troubleshoot possible problems, it's necessary to be familiar with the process architecture of Oracle Internet Directory.

Almost all documents say that tool called "oidctl" is used to start and stop OID servers, oidldapd (LDAP server) and oidrepld (replication server). This statement is slightly misleading, as oidctl doesn't directly control any of those.

When oidctl is executed, it connects to the database as user ODSCOMMON and simply inserts/updates rows into a table ODS.ODS_PROCESS depending on the options used in the command. A row is inserted if the START option is used, and updated if the STOP or RESTART option is used. So there are no processes started at this point, and LDAP server is not started.

In table ODS.ODS_PROCESS, we have the following information (list not complete):

```
- instance - the number of instance in question, must be unique
- pid      - process id, will be updated by oidmon when process is
             started
- state    - type of the operation requested
             possible values for state:
         0=stop
         1=start
         2=running
         3=restart
```

To control the processes (servers) we need to have OID Monitor (oidmon) running. This monitor is often called daemon or guardian process as well. When oidmon is running, it periodically connects to the database and reads the ODS.ODS_PROCESS table in order to start/stop/restart related processes.

When oidmon finds a row with state=0, it reads the pid and stops the process.

```
With state=1: oidmon starts a new process and updates pid with a new
              process id.

With state=2: oidmon reads the pid, and checks that the process with
              the same pid is running. If it's not, oidmon starts a new
              process and updates pid accordingly.
With state=3: oidmon reads the pid, stops the process, starts a new one
              and updates the pid accordingly. If oidmon can't start
              the server for some reason, it retries 10 times, and if
              still unsuccessful, it deletes the row from the
```

```
ODS.ODS_PROCESS table.
```
So oidctl only inserts/updates state information, and oidmon reads rows from ODS.ODS_PROCESS, and performs specified tasks based on the value of the state column.

In order to successfully troubleshoot possible problems related to start / stop of servers, we also need to know what processes are involved, and who controls them. So far, we have only discussed about a "process". In fact, the LDAP server contains at least two processes, based on the configuration used to start the server.

Oidmon itself is a process (called oidmon on unix, oidmon.exe on windows). When oidctl is used to start the server, we have to specify an instance number, which is any number between 0 and 1000. When oidmon starts this instance, it actually starts one process, which is the dispatcher/listener process. Note that this listener process is not the same as the net8 listener process. The id of this process is stored in the ODS.ODS_PROCESS table. Then that new process starts a number of server processes. This number is defined in the configuration set.

**NOTE:** These processes are started and controlled by the listener/dispatcher process, not by oidmon. If one of these processes dies for some reason, it's automatically restarted by the listener/dispatcher. Both the listener/dispatcher process and server process are called oidldapd on unix, and oidldapd on NT.

Similarly, when oidctl is used to stop the server, oidmon actually stops the listener/dispatcher process, which first stops all related server processes.

***Troubleshooting***

As we can notice, the architecture is fairly complicated, and therefore it's extremely important to understand how the product works before we can troubleshoot possible problems.

**Problems with oidctl**

First of all, syntax obviously has to be correct. Note:125301.1 provides a good set of examples for oidctl and oidmon. For detailed information, see the Oracle Internet Directory Administrator's Guide.

Because the only task oidctl has is to insert / update table ODS.ODS_PROCESS in the database, it's obvious that the database and listener have to be fully accessible when oidctl is used. On the other hand, error messages received are very clear if the DB/listener are not accessible.

One common cause of problem is user ODSCOMMON, which is used to connect to the database. If error ORA-1017 is signalled, it's worth checking that ODSCOMMON user has been created. This normally means an incomplete install. This can be fixed by reinstalling the product, or following steps listed in note:159031.1.

Also, the default password for ODSCOMMON is ODSCOMMON, and that cannot be changed. It's hardcoded in oidctl, and changing that on the database level will cause ORA-1017. Note that there's no security risk not being able to change ODSCOMMON password, as it has only connect privilege by default. Other privileges come via a role ODS_SERVER, which is password protected, and that password can be changed. Another common mistake is

the "connect" option in the oidctl syntax. The value of that option is the tns alias (connect string) to the database, not the hostname or anything else.

The easiest way to test that the database and net8 configuration are fine, and that all database components are like they should be, is to connect to the database with sqlplus installed in the same oracle home directory as oidctl, and login as odscommon/odscommon@<tns alias>, where <tns alias> is the same as used with the "connect" option in oidctl. Also, ensure that the database is the right one, not another one with OID installed.

When all this is working fine, then selecting from ODS.ODS_PROCESS should give rows with states described above.

**Processes don't start**

When confirmed that information in ODS.ODS_PROCESS is what it should be, and the problem still exists, we need to investigate why processes are not started.

First, like mentioned above, when everything is working fine, we should see at least three processes. One called oidmon, and at least two called oidldapd.

If oidmon is not running, there's no one to start/stop servers, and even if info in the table is correct, processes don't start. Also note that oidmon reads the ODS.ODS_PROCESS table using an interval, which is controlled by the "sleep" option when starting oidmon (default 10 secs). Always give some time for oidmon to complete the requested operation before stopping it. Also note that when oidmon is started, it doesn't connect to the database directly. Database connections are done periodically when oidmon is running. Therefore a problem with Net8 or the database itself will not cause ANY errors to be signalled at the time of oidmon startup. Everything seems to be ok, but oidmon process disappears. See oidmon.log for details about the problem.

Although documentation says that oidmon must be started before oidctl is used, this is not mandatory because oidmon and oidctl don't directly communicate with each other. This can cause scenarios, where everything is stopped (=no processes running) after a machine reboot, but using oidctl to start an instance gives an error saying that the specified instance number is already in use.

If oidmon is running fine, and information is correct in the ODS.ODS_PROCESS table, and still processes don't start or connect to the LDAP server fails, we need to take a look to traces generated.

All traces are created in the directory $ORACLE_HOME/ldap/log (Unix) or %ORACLE_HOME%\ldap\log (Windows), and use the following naming format:

```
- oidmon.log
- oidldapd<xx>.log       where <xx> is the instance number
- oidldapd<xx>s<yy>.log  where <xx> is the instance number and <yy>
                         is the pid.
```

oidldapd<xx>.log is created by thelistener/dispatcher process (one per instance) and oidldapd<xx>s<yy>.log by the server process(at least one per instance).

oidmon.log doesn't normally give useful information, as oidmon doesn't know why a process is not started, or why it's dying. You will probably only see information which tells that the process is not running,restarting process.

But for troubleshooting, traces created by listener/dispatcher and server processes are relevant.

If the error listed in the trace doesn't give any hits in metalink, the following should be done:

```
- shutdown LDAP the servers and oidmon, if running (on Windows, stop
  the directory service as well)
- remove/rename old trace files
- start oidmon and the LDAP server with maximum debug level 65535.
  Note that you need to stop/start the server in order to get the
  trace,restarting is not enough (see bug:1702226)
- investigate new traces, and if needed, log an iTAR with Oracle
  Support Services and upload all traces to the iTAR.
```

**Known problems on this area**

```
Bug:1816256  OIDLDAPD PROCESSES ARE NOT KILLED WHEN DB IS SHUTDOWN/CRASHES
Bug:1608778  LDAP SERVER FAILOVER DOES NOT WORK
Bug:1940996  LDAP SERVER DOESN'T START IF FLAGS ARE USED
```

***Related Documents***

```
Oracle Internet Directory Administrator's Guide
Note:121997.1     Unable to connect to OID Server - Bind Failed
Note:91435.1      Cannot Start LDAP instance
Note:1015431.102  ORA-1000 WHILE ADDING ENTRY CN=INSTANCE1,CN=OSDLDAPD,
                  CN=SUBREGISTRYSUBENTRY
```

---

## Starting and Stopping OID

*by Jeff Hunter, Sr. Database Administrator*

---

**Contents**



---

### The OID Monitor Process

The OID Monitor must be running to process commands to start and stop the server.

OID Monitor is a component that initiates, monitors, and terminates the Oracle directory server processes.

It also controls the replication server if one is installed, and the Oracle directory integration server.

**The Server Instances**
The OID Control Utility, "**oidctl**" is a command-line tool for issuing run-server and stop-server commands.

The commands are interpreted and executed by the **OID Monitor** process.

**Scripts used to Start/Stop the directory services**
I created two scripts that can be used to start and stop the Oracle Internet Directory Server:

- **start_OID.sh**

  Starts the OID Monitor Process as well as any "oidldap" Server Instances. This performs the following tasks:

  - Switches the Oracle Environment to the ORACLE_HOME of the OID installation.
  - Starts the OID Monitor Process:

    ```
    % oidmon connect=OIDDB start
    ```

  - Starts any Server Instances using the OID Control Utility:

    ```
    % oidctl connect=OIDDB server=oidldapd instance=1 start
    ```

- **stop_OID.sh**

  Stops the OID Monitor Process as well as any "oidldap" Server Instances. This performs the following tasks:

  - Switches the Oracle Environment to the ORACLE_HOME of the OID installation.
  - Stops any Server Instances using the OID Control Utility:

    ```
    % oidctl connect=OIDDB server=oidldapd instance=1  stop
    ```

  - Stops the OID Monitor Process:

    ```
    % oidmon connect=OIDDB stop
    ```

---

## Securing the Oracle Internet Directory After Installation

*by Jeff Hunter, Sr. Database Administrator*

---

**Contents**

---

## Overview

After installing and configuring Oracle Internet Directory (OID), the DBA should take the proper precautions in securing all privileged accounts within the database and directory.

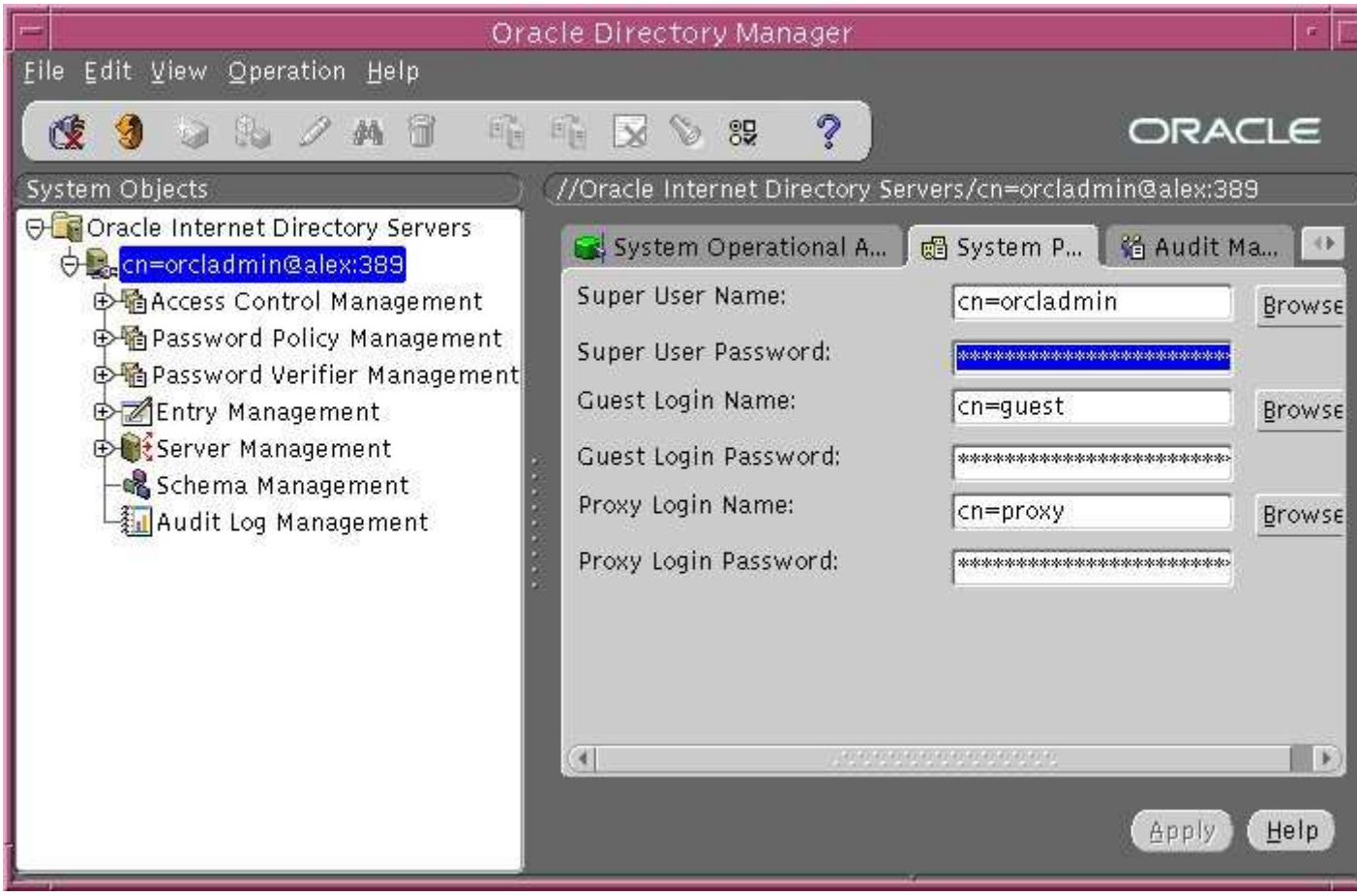
This article will present the steps necessary to secure all privileged accounts both in the LDAP directory and the Oracle database. For complete instructions for installing Oracle Internet Directory (Version 9.2.0), see my article entitled: "Installing Oracle Internet Directory - (Version 9.2.0)".

## Changing Directory Passwords

As part of the default DIT creation, several highly privileged users are created that should be secured before putting the directory into production.

### cn=orcladmin

One of the first accounts to secure is the Oracle Internet Directory Super User, **`cn=orcladmin`**. I generally use the GUI Java application, *Oracle Directory Manager*. First, login to the Oracle Directory Manager application as the Directory Super User (`cn=orcladmin`). The default password for this user is `welcome`. After successfully logging into the directory, use the navigation menu (the tree menu on the left) and click on the entry directly below the very top entry (Oracle Internet Directory Servers). This entry will be the one that is your current connection with the LDAP directory. For my example, this entry is labeled: "`cn=orcladmin@alex:389`" as show below. After clicking on this entry, you will be presented with a *tabbed* window in the content pane (the right pane). From here, click on the tab named "System Passwords". To change the password of the Directory Super User, click on the password text field named "*Super User Password*", change the password and click the "Apply" button.




## Oracle Database Passwords

### OID Database Schema Owner

The Oracle Internet Directory runs on an Oracle database and creates two database users: **ODS** and **ODSCOMMON**. ODS is the schema owner that contains all of the database objects (tables, views, objects, etc.) used for OID functionality and directory storage. When the OID needs to login to the database, it uses the ODS database account which has a default password of ODS. You should secure this database user account before putting the LDAP directory into production.

### Using the OID Database Password Utility

The DBA can change this password by using the *OID Database Password Utility* (included with the OID installation). The following example uses this utility to change the database password for ODS:

```
# $ORACLE_HOME/bin/oidpasswd
current password: ods
    new password: new_secret_password
confirm password: new_secret_password
password set.
```

### How oidpasswd Works

The oidpasswd utility connects as the **ODSCOMMON** user and uses the role **ODS_SERVER** with the original password to perform the following:

1. Changes the password for the ODS user for the OID schema database.
2. Updates the SYSTEM.ODSINSTANCES table with the new, encrypted password for ODS.
3. Changes the password for the ODS_SERVER role to the new password.

4. Updates the `$ORACLE_HOME/ldap/admin/oidpwdr` file with the encrypted password.

**NOTE:** Some Oracle OID patches and scripts may assume the user `ODS/ODS`. It is advised to change the password for `ODS` back to its default of `ODS` during application of patches or when running OID scripts.

**But what about `ODSCOMMON`?**

When connecting to the database schema, the OID executables, such as `oidctl` or `oidldapd` servers connect as the database user `ODSCOMMON`. The password for `ODSCOMMON` is `ODSCOMMON`, and that password cannot be changed. It is hardcoded in the executables and changing it on the database level will cause an *ORA-01017* error. There is no security risk, however, not being able to change the `ODSCOMMON` password, as it has only CONNECT privilege. Once connected as `ODSCOMMON`, the executable will obtain the privileges it needs via the role `ODS_SERVER`, which is protected by the `ODS` password. The password is obtained by the executable from the `SYSTEM.ODSINSTANCES` table. This is the password encrypted and set by the `oidpasswd` utility.

---

## Checking OID/LDAP Operation

*by Jeff Hunter, Sr. Database Administrator*

---

This article provides a means to quickly identify if the Oracle Internet Directory (LDAP) service is installed and functioning properly.

### UNIX

If have Oracle Internet Directory installed on the UNIX platform, use the command-line tool **`ldapcheck`** as follows:

```
# $ORACLE_HOME/ldap/bin/ldapcheck

Checking Oracle Internet Directory Processes ...

Process oidmon is Alive as PID 401      <--- guardian process
Process oidldapd is Alive as PID 444    <--- ldap dispatcher
Process oidldapd is Alive as PID 438    <--- ldap server
Not Running ---- Process oidrepld       <--- Only shows if OID Replication
was setup
```

### Windows

Instead of using `ldapcheck`, use the *task manager* and you should see:

```
oidservice.exe   <--- the equivilent of oidmon on unix
oidldapd.exe     <--- two instances, the dispatcher and server..
```

---

# Default LDAP Port Numbers Not Working

*by Jeff Hunter, Sr. Database Administrator*

---

This article provides an insight into some of the problem that you may encounter with default LDAP port numbers when installing Oracle Internet Directory (OID) 9.0.2 and higher. If you are a user of OID version 2.1.1 or 3.0.1, you will not have this problem. For those running OID version 9.0.2 and higher, it is possible that the installer will choose ports other than the default (i.e. 389) due to the current system environment at installation. If the installer does choose another port other than the default, the LDAP tools that are provided with OID will not work without providing a port number as a command-line argument.

If you are among those unfortunate users that attempts to run any of the LDAP tools provided with OID (i.e. `ldapbind`) and get a message like "*Cannot connect to the LDAP server*", this is a good indication that your LDAP directory server is either down or not listening on the default ports that the LDAP tools expect. It is very possible that the LDAP Directory port is 4032 (or another port as chosen by the installer), but not 389. If this is the case, you must add the following option everytime with all OID command line LDAP tools:

```
-p <port number>
```

For example, if the port chosen by the installer is 4032 and the server name is `dbautil`, you would have to run the commands as follows:

```
ldapbind -h dbautil -p 4032
bind successful

ldapsearch -h dbautil -p 4032 -b "" -s base "objectClass=*"
orcldirectoryversion=OID 9.2.0.1.0
supportedcontrol=2.16.840.1.113730.3.4.2
supportedcontrol=2.16.840.1.113894.1.8.1
supportedcontrol=2.16.840.1.113894.1.8.2
supportedldapversion=LDAP Version 2
supportedldapversion=LDAP Version 3
subschemasubentry=cn=subschemasubentry
subconfigsubentry=cn=subconfigsubentry
subregistrysubentry=cn=subregistrysubentry
changelog=cn=changelog
changestatus=cn=changestatus
orclservermode=rw
orclauditlevel=0
orclsuname=cn=orcladmin
orclsupassword={MD4}daHKLtgPBN1v54JGCyEBVg==
orcldebugflag=0
orclanonymousbindsflag=1
orcloptcontainsquery=0
orclprname=cn=proxy
orclprpassword={MD4}u/7/MSiUpRFIG2FergLc7w==
orclguname=cn=guest
orclgupassword={MD4}KzLGjQo0wbA9u9v7On6Kmg==
orclreplagreements=cn=orclreplagreements
matchingrules=distinguishedNameMatch
matchingrules=caseIgnoreMatch
matchingrules=caseExactMatch
matchingrules=numericStringMatch
matchingrules=telephoneNumberMatch
```

```
orclcatalogentrydn=cn=catalogs
orclsizelimit=1000
orcltimelimit=3600
orclenablegroupcache=1
orclecacheenabled=1
orclecachemaxsize=100000000
orclecachemaxentries=25000
orclmatchdnenabled=1
orclupgradeinprogress=FALSE
orclcryptoscheme=MD4
orclstatsflag=0
orclstatsperiodicity=60
orclstatslevel=0
orcldiprepository=FALSE
orclreplicaid=dbautil
orclaci=access to entry by * (browse,noadd,nodelete)
orclaci=access to attr=(userpkcs12,orclpkcs12hint,userpassword) by
group="cn=OracleUserSecurityAdmins,cn=Groups,cn=OracleContext"
(search,read,write,compare) by self (search,read,write,compare) by * (none)
orclaci=access to attr=(orclpassword) by self (search,read,write,compare)
by * (none)
orclaci=access to attr=(orclpasswordverifier) by self
(search,read,write,compare) by * (none)
orclaci=access to attr=(orclstatsflag, orclstatsperiodicity) by dn="cn=emd
admin,cn=oracle internet directory" (search,read,write,compare) by *
(search,read)
orclaci=access to
attr!=(userpkcs12,orclpkcs12hint,userpassword,orclpassword,orclpasswordveri
fier,orclstatsflag,orclstatsperiodicity) by * (search,read,compare)
```

The command line tools that come with OID, which comply with the LDAP RFC standard, will not be changed to use a different port as a default.

Starting with OID version 9.0.2, the installer has allocated a range of ports for each product apart from the product default ports. For OID, if the default ports which are 389 (non-SSL), and 636 (SSL) is not available, the installer will get a free port in the range: 4031 to 4039.

The installer determines if port 389 is free based on the following conditions:

- Port 389 is free
- There is no line with port 389 in the `/etc/services` file (which indicates that this port is reserved for some service).

---

## How to Change the OID Ports

*by Jeff Hunter, Sr. Database Administrator*

---

**Changing the OID Port Numbers**

This article explains how to change the port numbers used by the Oracle Internet Directory from those assigned at the time of installation. Keep in mind that this article only refers to version 9.0.1, 9.0.2, and 9.2.0.

It may be deemed necessary at some point to change the ports OID is using from those assigned when OID was installed. The best way to accomplish this is to create a new *Configuration Set* with the new ports. You would then start the OID processes using the new Configuration Set. The following is a list of the steps you can use to create a new Configuration Set:

1. Login to the Oracle Directory Manager.
2. Navigate to *Server Manager -> Directory Server*
3. Right-click on *Default Configuration Set* and select *Create Like*. The new configset number is automatically incremented.
4. On the *General* tab, edit the Non SSL Port as desired.
5. On the SSL Settings tab, edit the SSL Port as desired.
6. Click OK to save the new configset. Ensure it is now listed under the previous one.
7. Exit from the Oracle Directory Manager.
8. Shutdown OID.
9. Restart OID using the new configset:
10. `# oidmon connect=<Service Name> start`
    `# oidctl connect=<Service Name> server=oidldapd instance=<New Configset Number> start`

11. Launch Oracle Directory Manager.
12. Click on the icon to the right of the *Server* field on teh login screen.
13. Select the existing entry for the OID server and click on *Edit*.
14. Update the new port, then click OK.
15. Login to Oracle Directory Manager to ensure OID is up and listening on the new port.

## Oracle9iAS Infrastructure Users

For those using Oracle9*i*AS Infrastructure OID, this change will also require changes to a number of 9*i*AS configuration files, both on the infrastructure and for any midtier servers.

1. Run the following SQL script to update the SSO configuration:

   UNIX: **`$ORACLE_HOME/sso/admin/plsql/sso/ssooconf.sql`**
   Windows: **`%ORACLE_HOME%\sso\admin\plsql\sso\ssooconf.sql`**

   If only the <ENTER> is pressed in response to all questions then the configuration remains unchanged and the current values are displayed.

   Support the new OID port numbers when prompted. The output looks like the following:

   ```
   ------------------------------------------------
   CONFIGURATION FOR SSO SERVER:
   LDAP HOST: dbautil.idevelopment.info
   ```

```
LDAP PORT: 4032
SSO SERVER DN:
orclApplicationCommonName=ORASSO_SERVER,cn=SSO,cn=Products,cn=O
racleContext
SSO SERVER PASSWORD: 6C189B1EF32BF7182A90A1C9358E
LDAP USE SSL: N
-----------------------------------------------
```

2. Edit the `ias.properties` file: UNIX:
   **`$ORACLE_HOME/config/ias.properties`**
   Windows: **`%ORACLE_HOME%\config\ias.properties`**

   Under the section: `[InstallData]` change `OIDport` to the new port. e.g.

   ```
   [InstallData]
   ...
   OIDhost=dbautil.idevelopment.info
   OIDport=4032
   ```

3. Check the documentation for each midtier to determine how to update the appropriate configuration files. Note that the following components will definately be affected: JAZN, Forms, Reports, Portal, Discover, OLAP, Oracle Colaboration Suite.

---

## Using LDAP Command-Line Tools

*by Jeff Hunter, Sr. Database Administrator*

---

**Contents**



---

**Overview**

The installation of Oracle Internet Directory will include several command-line tools that can be useful for searching and modifying entries within your LDAP directory. Some of these utilities include:

- **`ldapadd`**
- **`ldapsearch`**

- **ldapmodify**
- **ldapdelete**
- **ldapmoddn**
- **bulkmodify**
- **ldapcompare**

For the purpose of this document, I will be providing an example of how to add, search, modify, and delete an LDAP entry. The directory context (document root) I will be using for these example is `o=airius.com`. For details on configuring your LDAP directory with the `airius.com` examples, see my article on: Importing airiusplus.ldif into Oracle Internet Directory.

## Adding an Entry

In this section, I will be adding an entry to an LDAP directory using the `ldapadd` command.

### Creating an LDIF File

Create an LDIF file (as shown below) named `melody.ldif`:

```
dn: uid=mhunter, ou=People, o=airius.com
givenname: Melody
telephonenumber: +1 412 555 8234
sn: Hunter
userpassword: {MD4}yLYn5mv9vZ1kq/hHfuiFCg==
ou: Human Resources
ou: People
l: Butler
roomnumber: 1213
manager: uid=jhunter, ou=People, o=airius.com
mail: mhunter@airius.com
facsimiletelephonenumber: +1 412 555 8235
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
uid: mhunter
cn: Melody Hunter
title: VP, Operations
```

Assuming the host for the directory server is named `alex` and running on port 389, you can use the following to add the above entry:

```
# ldapadd -v -D "cn=orcladmin" -w "welcome" -p 389 -h alex -f melody.ldif
add givenname:
        Melody
add telephonenumber:
        +1 412 555 8234
add sn:
        Hunter
add userpassword:
        {MD4}yLYn5mv9vZ1kq/hHfuiFCg==
add ou:
        Human Resources
        People
add l:
        Butler
add roomnumber:
```

```
        1213
add manager:
        uid=jhunter, ou=People, o=airius.com
add mail:
        mhunter@airius.com
add facsimiletelephonenumber:
        +1 412 555 8235
add objectclass:
        top
        person
        organizationalPerson
        inetOrgPerson
add uid:
        mhunter
add cn:
        Melody Hunter
add title:
        VP, Operations
adding new entry uid=mhunter, ou=People, o=airius.com
modify complete
```

**Using a Here Document**

You can also use the `ldapadd` without having to first create the file by using a here document as follows:

```
# ldapadd -D "cn=orcladmin" -w "welcome" -p 389 -h alex <<EOF
> dn: uid=mhunter, ou=People, o=airius.com
> givenname: Melody
> telephonenumber: +1 412 555 8234
> sn: Hunter
> userpassword: {MD4}yLYn5mv9vZ1kq/hHfuiFCg==
> ou: Human Resources
> ou: People
> l: Butler
> roomnumber: 1213
> manager: uid=jhunter, ou=People, o=airius.com
> mail: mhunter@airius.com
> facsimiletelephonenumber: +1 412 555 8235
> objectclass: top
> objectclass: person
> objectclass: organizationalPerson
> objectclass: inetOrgPerson
> uid: mhunter
> cn: Melody Hunter
> title: VP, Operations
> EOF

adding new entry uid=mhunter, ou=People, o=airius.com
```

**NOTE:** Keep the following in mind when attempting to add an entry to your LDAP directory:

- You cannot enter any operational attributes *(i.e. orclguid, creatorsname, modifiersname, createtimestamp, modifytimestamp, pwdchangedtime)* when attempting to add an entry.
- They `-v` command-line option specifies verbose mode which simply means to give a detailed output from the command.
- You cannot add a new entry (a non-first level entry) without a parent. Attempting

to do so will result in an error. If I try to add, for example, an entry to `ou=People2, o=airius.com` (which does not exist), I will get the following error:

- `adding new entry uid=mhunter, ou=People2, o=airius.com`
  `ldap_add_s: No such object`

## Searching for an Entry

Now that we have successfully added an entry, I want to search for it. For this, we can use the command-line tool `ldapsearch`. Here is the general syntax for the `ldapsearch` command that is included with Oracle Internet Directory:

```
ldapsearch [options] filter [attributes...]
```

---

In the following query, I want to search for the entry we just added and include all of the attributes:

```
# ldapsearch -v -D "cn=orcladmin" -w "welcome" -h alex -p 389 -s sub -b
"o=airius.com" "uid=mhunter"
ldap_init( alex, 389 )
filter pattern: uid=mhunter
returning: ALL
filter is: (uid=mhunter)
uid=mhunter, ou=People, o=airius.com
givenname=Melody
telephonenumber=+1 412 555 8234
sn=Hunter
userpassword={MD4}yLYn5mv9vZ1kq/hHfuiFCg==
ou=Human Resources
ou=People
l=Butler
roomnumber=1213
manager=uid=jhunter, ou=People, o=airius.com
mail=mhunter@airius.com
facsimiletelephonenumber=+1 412 555 8235
objectclass=top
objectclass=person
objectclass=organizationalPerson
objectclass=inetOrgPerson
uid=mhunter
cn=Melody Hunter
title=VP, Operations
1 matches
```

---

Now, I want to perform the same search, but this time not include the verbose option (`-v`) and only include several of the attributes for the output:

```
# ldapsearch -D "cn=orcladmin" -w "welcome" -h alex -p 389 -s sub -b
"o=airius.com" "uid=mhunter" dn cn uid
uid=mhunter, ou=People, o=airius.com
cn=Melody Hunter
uid=mhunter
```

---

For this search, I want to limit the amount of entries being returned. To do this, I can use the `-z` parameter. If I were to search for all entries with `Jeff` as part of the command name (`cn:`), I would have 4 records. For this example, I want to limit the search to only two records:

```
# ldapsearch -D "cn=orcladmin" -w "welcome" -h alex -p 389 -s sub -b
"o=airius.com" -z 2 "cn=Jeff*" dn cn uid
uid=jcampai2, ou=People, o=airius.com
cn=Jeffrey Campaigne
uid=jcampai2

uid=jmuffly, ou=People, o=airius.com
cn=Jeff Muffly
uid=jmuffly
ldap_search: Sizelimit exceeded
```

## Modifying an Entry

The Oracle Internet Directory includes the `ldapmodify` utility that, like its name suggests, is used to modify entries stored within the LDAP directory. Although a book could be written on just using the `ldapmodify` command, I will provide a few examples that will give you a feel for how this command can be used.

---

To start off, here is an example that adds an optional attribute to the entry we added in the `ldapadd` example above. Let's add the `departmentNumber` attribute by first creating an LDIF file (shown below) named `melody_modify.ldif`:

```
dn: uid=mhunter, ou=People, o=airius.com
changetype: modify
add: departmentNumber
departmentNumber: HR-342
```

Assuming the host for the directory server is named `alex` and running on port 389, you can use the following to modify the above entry in order to add the `departmentNumber` attribute:

```
# ldapmodify -D "cn=orcladmin" -w "welcome" -p 389 -h alex -f
melody_modify.ldif
modifying entry uid=mhunter, ou=People, o=airius.com
```

---

Now let's use the *delete* option for `ldapmodify` in order to remove the optional attribute we just added: `departmentNumber`. Just like above, we create an LDIF file named `melody_modify.ldif` and use the `ldapmodify` command as follows:

```
dn: uid=mhunter, ou=People, o=airius.com
changetype: modify
delete: departmentNumber
```

```
# ldapmodify -D "cn=orcladmin" -w "welcome" -p 389 -h alex -f
melody_modify.ldif
modifying entry uid=mhunter, ou=People, o=airius.com
```

---

For a last example, let's use the *replace* option of the `ldapmodify` command. For this example, I want to replace the title from *"VP, Operations"* to *"VP, Human Resources"* as follows:

```
dn: uid=mhunter, ou=People, o=airius.com
changetype: modify
replace: title
```

```
title: VP, Human Resources
```

```
# ldapmodify -D "cn=orcladmin" -w "welcome" -p 389 -h alex -f
melody_modify.ldif
modifying entry uid=mhunter, ou=People, o=airius.com
```

## Deleting an Entry

One of the last type of operations you would typically want to perform on an LDAP entry is to delete it. This is where the `ldapdelete` command-line tool can come in handy. Keep in mind, however, that you can only delete non-leaf entries. If you try to delete a non-leaf entry, the operation will fail. Let's know delete the entry we created in the `ldapadd` section of this document:

```
# ldapdelete -D "cn=orcladmin" -w "welcome" -p 389 -h alex -v "uid=mhunter,
ou=People, o=airius.com"
deleting entry uid=mhunter, ou=People, o=airius.com
delete completed
```

## Modify RDN / DN Operations

One of the last commands I will be covering in this article is the `ldapmoddn` command. You can use this command to modify any RDN or DN entry you have access to.

---

For this first example, I will modify an RDN of the following entry: "`uid=mlott, ou=People, o=airius.com`".

```
# ldapmoddn -D "cn=orcladmin" -w "welcome" -p 389 -h alex -b "uid=mlott,
ou=People, o=airius.com" -R "uid=mikelott"
uid=mlott, ou=People, o=airius.com renamed successfully.
```

---

Now, for a more complex example. I want to change (or better said, move the RDN "`ahunter`") the entry "`uid=ahunter, ou=People, o=airius.com`" to "`uid=ahunter, ou=People, dc=idevelopment, dc=info`":

```
# ldapmoddn -D "cn=orcladmin" -w "welcome" -p 389 -h alex -b "uid=ahunter,
ou=People, o=airius.com" -R "uid=ahunter" -N "ou=People, dc=idevelopment,
dc=info"
uid=ahunter, ou=People, o=airius.com renamed successfully.
```

---

# Configuring Oracle Net8 with LDAP

*by Jeff Hunter, Sr. Database Administrator*

---

## Contents

5. **Creating a Net Server Name**

---

## Overview of Net8 and LDAP

Beginning with Release 8.1.6, Oracle has built support into Net8 for the use of LDAP as a name resolution method. LDAP is now preferred over Oracle Names in cases where you need a centralized repository for net server names.

One of the great things about LDAP is that you can create your own object classes and attributes. This allows you to use LDAP directories for a wide variety of creative purposes. Oracle currently supports LDAP for the following uses:

- Global Users
- Global Roles
- Net Server Names

Global users and global roles are defined in an LDAP directory service and can be managed centrally. A user can change his password once, for example, and have that change apply to all databases across the board. Similarly, net server names may also be defined and managed separately. Oracle's clear goal is to take all bits and pieces of information that DBAs usually need to replicate for each database, and allow those to be centrally managed. LDAP is the core technology supporting this effort, and it will play an increasingly important role in our Oracle environment.

## Configuring Net8 Clients for LDAP

### Specify the LDAP Naming Method - (sqlnet.ora)

You specify naming methods through the `NAMES.DIRECTORY_PATH` parameter in your `sqlnet.ora` file. The keyword for directory naming is **`LDAP`**. The following parameter setting will configure a client to attempt name resolution through the local `tnsnames.ora` file first, and then through the LDAP directory.

```
NAMES.DIRECTORY_PATH= (TNSNAMES, LDAP)
```

### Addressing an LDAP Server - (ldap.ora)

Once you've specified LDAP as a naming method, you need to identify an LDAP directory server for the client to contact. You do this in the ldap.ora file using the two parameters `DIRECTORY_SERVERS` and `DIRECTORY_SERVER_TYPE`.

DIRECTORY_SERVERS

The `DIRECTORY_SERVERS` parameter specifies the network address of one or more LDAP directory servers. A directory server address consists of a hostname and two port numbers. The first port number is used to unsecured connections. The second port number is optional and is used for SSL (Secured Socket Layer) connections. The following example illustrates the default port numbers that OID uses for the two connection types:

```
DIRECTORY_SERVERS= (ldap.idevelopment.info:389:636)
```

DIRECTORY_SERVER_TYPE

The DIRECTORY_SERVER_TYPE parameter identifies the brand of LDAP server that you are using. The following three values represent valid types:

- **OID** - Oracle Internet Directory
- **AD** - Microsoft Active Directory
- **NDS** - Novell Directory Services

The following is an example setting that uses Oracle Internet Directory as the LDAP server:

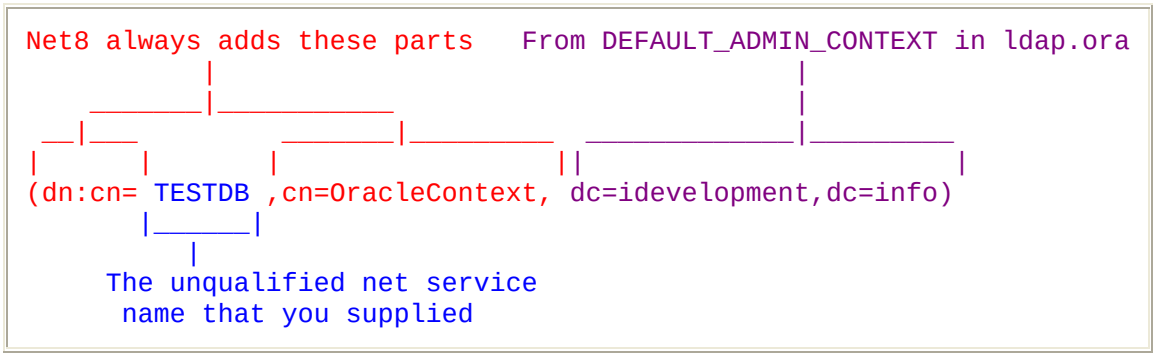
```
DIRECTORY_SERVER_TYPE = OID
```

**NOTE:** Net8 does not support mixing LDAP directory products. The DIRECTORY_SERVER_TYPE setting applies to all the directory server addresses listed for the DIRECTORY_SERVERS parameter.

**Specify a Default Administrative Context - (ldap.ora)**

The default administrative context is the LDAP equivalent of the default Net8 domain. LDAP directory structures do not necessarily need to correspond to any sort of domain name structure, so a new mechanism is needed to specify the context in which unqualified net server names are resolved. This mechanism is the default administrative context, which you specify using the DEFAULT_ADMIN_CONTEXT parameter in your ldap.ora file. For example:

```
DEFAULT_ADMIN_CONTEXT = "dc=idevelopment,dc=info"
```




*Net8 creates a full distinguished name from a simple unqualified net server name*

Once Net8 has translated an unqualified net server name into a distinguished name, it passes that name to the LDAP directory to be used. The LDAP directory then returns the definition of the name, giving Net8 information it needs to make the connection to the appropriate database service.

**NOTE:** The NAMES.DEFAULT_DOMAIN parameter in sqlnet.ora is ignored when directory naming is used. Instead, equivalent functionality (implemented in a manner suitable to LDAP) is provided through the DEFAULT_ADMIN_CONTEXT parameter in ldap.ora.

**Defining Net Server Names in an LDAP Directory**

As a prerequisite to using LDAP for net server name resolution, you need to have the Oracle Internet Directory installed. You also need to have the LDAP *schema* for Net8 in place. This schema comprises the LDAP object classes on which the entries defining net service names are based. The LDAP schema for Net8 is installed by default when you install OID.

**Create the Administrative Context**

All the examples in this section use an administrative context based on DNS. The domain used is `idevelopment.info`, which requires the following LDAP entry:

```
(dn: dc=idevelopment, dc=info)
```

Underneath that entry, Net8 expects to find the `cn=OracleContext` entry. Using Oracle Directory Manger, locate the folder named "*Entry Management*" and add the following entries:

```
dn: dc=info
objectclass: domain
objectclass: top
dc: info

dn: dc=idevelopment,dc=info
objectclass: domain
objectclass: top
dc: idevelopment
```

**Create the OracleContext entry**

```
dn: cn=OracleContext,dc=idevelopment,dc=info
objectclass: orclContext
objectclass: top
cn: OracleContext
```

## Create an LDAP User to Manager Net Server Names

### Creating LDAP User

If you have only one *admin context*, you can create the directory user underneath that context. You may want to create an additional entry named **`OracleNetAdmins`**, and collect all the Net8 directory users underneath that. I typically use the entry **`OracleNetAdmins`** to store this user information.

A user is simply another entry in an LDAP directory, but it is based in part on the object class named *person*.

```
dn: cn=OracleNetAdmins,cn=OracleContext,dc=idevelopment,dc=info
objectclass: orclContext
objectclass: top
cn: OracleNetAdmins

dn: cn=jhunter,cn=OracleNetAdmins,cn=OracleContext,dc=idevelopment,dc=info
userpassword: {MD4}4AC9DBA1A08FC124DD57C915D540BD3F
objectclass: orclContext
objectclass: top
objectclass: person
cn: jhunter
sn: Jeffrey Hunter
```

**NOTE:** The person object class comes with two mandatory attributes, cn and sn, that you should fill in with the user's login name and full name, respectively. The password attribute is not mandatory and is consequently found under the Optional Properties tab. Because of its placement, it's easy to overlook the password. When creating a user, be sure that you go to the Optional Properties tab and specify a password.

### Granting Access to LDAP User

After creating a directory user, you must give that user some access rights. If you want the user to be able to manager net server names that fall under dc=idevelopment,dc=info, then

you must grant the user access to that part of the LDAP directory tree. To do that, highlight the entry for dc=pit in the ODM's left pane, and click the tab in right pane title Subtree Access. You can then grant a user access to that entry, and on all those entries that fall beneath it.

In the dialog box on the right under Subtree Access, make sure to choose the "Structural Access Items". (The top box). Click the "Create" button under "Structual Access Items".

Use the "By Whom" tab and enter the name of the user to grant access to:

```
cn=jhunter,cn=OracleNetAdmins,cn=OracleContext,dc=idevelopment,dc=info
```

After identifying the user, you need to specify the rights that you are granting to that user. You do that from the "Access Rights" tab. Here you would select to grant:

```
Browse, Add and Delete
```

After creating a user and granting access rights, you can use the Java application "*Net Manager*" to log in to the LDAP directory and create net service names definitions.

**Creating a Net Server Name**

The easiest way to define net service names in an LDAP directory is to use the Oracle application "*Net Manager*".

---

## Importing `airiusplus.ldif` into Oracle Internet Directory

*by Jeff Hunter, Sr. Database Administrator*

---

**Contents**



---

**Overview**

"LDAP Programming with Java" published by Addison Wesley Longman, Inc. is a complete guide about LDAP. Although the Netscape Directory Server (NDS) is used in this book, most of topics are concentrated in Java programming. Because I'm using Oracle Internet Directory instead of NDS on Linux (and Windows), I needed to make several adjustments to the configurations so that I can follow sections in title.

This short article attempts to provide some of the steps I needed to complete in order to import their sample database: "`airiusplus.ldif`".

## Downloading the `airiusplus.ldif` File

Use the following link to download the latest sample database (LDIF File): airiusplus.ldif

## Alter the `airiusplus.ldif` File

There are two changes that need to be made to the `airiusplus.ldif` data file. Before making modifications to the original file, make a backup copy first.

The first thing I needed to do was to get rid of any of the "**`aci:`**" attributes. All together, there will be seven "**`aci:`**" attributes contained within two DN's: **`dn: o=airius.com`** and **`dn: ou=People, o=airius.com`**. All seven "**`aci:`**" attribute settings are contained on multiple lines (using the LDIF line continuation standards). Ensure that you remove the entire attribute setting by deleting all lines of the "**`aci:`**" setting.

Next, change the values for all **`userpassword:`** attributes to include at least one numeric character. Failure to do so, will result in errors when attempting to import all People. If you are using `vi`, use the substitute syntax:

```
:%s/^userpassword: /userpassword: 23/
```

## Import the File

The Netscape Directory SDK for Java includes several tools that can be used to use with an LDAP directory:

- `LDAPDelete.java`
- `LDAPModify.java`
- `LDAPSearch.java`
- `LDAPTool.java`

For details about configuring and building these tools, refer to the article Building Netscape LDAP Tools for Java - (Version 4.1.7 / LDAPSearch, LDAPModify, etc.)

After locating / building the `LDAPModify` application, use the following to import the data:

```
# java LDAPModify -c -a -h ldap.idevelopment.info -D "cn=orcladmin" -w
"welcome" -f "airiusplus.ldif"
adding new entry o=airius.com

adding new entry ou=Groups, o=airius.com

adding new entry cn=Directory Administrators, ou=Groups, o=airius.com

adding new entry ou=People, o=airius.com

adding new entry ou=Special Users,o=airius.com

adding new entry uid=scarter, ou=People, o=airius.com

adding new entry uid=tmorris, ou=People, o=airius.com

adding new entry uid=kvaughan, ou=People, o=airius.com

. . . SNIP . . .

adding new entry uid=jvedder, ou=People, o=airius.com

adding new entry cn=Accounting Managers,ou=groups,o=airius.com
```

```
adding new entry cn=HR Managers,ou=groups,o=airius.com

adding new entry cn=QA Managers,ou=groups,o=airius.com

adding new entry cn=PD Managers,ou=groups,o=airius.com

adding new entry ou=Netscape Servers,o=airius.com
```

---

## Writing LDAP Entries to an LDIF File - *(Using ldapwrite and ldapsearch)*

*by Jeff Hunter, Sr. Database Administrator*

---

**Contents**



---

### Overview

There are two easy ways in which to write LDAP entries (application data) to an LDIF flat file. Those two ways are the command-line utilities **`ldifwrite`** or **`ldapsearch`**.

### Using `ldifwrite`

`ldifwrite` is intended for use only on OID "application data", not the schema or operational attributes and values (e.g., cn=catalogs, cn=changelog, etc.). Keep in mind that in order to use the command-line utility, you will need to know and use the `ODS` database username and password. The syntax for the `ldifwrite` command is:

```
ldifwrite -c <db_connect_string> -b <base DN> -f <filename>
```

### Exporting Application Data and Values

You can type the following commands to export OID application data using `ldapwrite`:

```
# ldifwrite -c OIDDB_ALEX -b "o=airius.com" -f airius.ldif
This tool can only be executed if you know database user password for OiD
Enter OiD Password :: ods
```

### Exporting Schema or Operational Attributes and Values

As previously mentioned, you cannot use `ldifwrite` to write out schema or operational data. In order to extract schema and operational data, you will need to use `ldapsearch` (described below). Trying to perform a search of this type, will result in the following error:

```
# ldifwrite -c OIDDB_ALEX -b "cn=subschemaSubentry" -f alex_oid_schema.ldif
This tool can only be executed if you know database user password for OiD
Enter OiD Password :: ods
Base DN cn=subschemasubentry not found.
```

### Using `ldapsearch`

The syntax for the `ldifsearch` command, while writing out to an LDIF file is:

```
ldapsearch -L -D "cn=orcladmin" -w "welcome" -h <host> -p <port> -b <base
DN> -s <Search Scope (base | one | sub)> "<Search Filter>" >
output_file.ldif
```

### Exporting Application Data and Values

You can type the following commands to export OID application data using `ldapsearch`:

```
# ldapsearch -L -D "cn=orcladmin" -w "welcome" -h alex -p 389 -b
"o=airius.com" -s sub "objectclass=*" > airius.ldif
```

### Exporting Schema or Operational Attributes and Values

As previously mentioned, you cannot use `ldifwrite` to write out schema or operational data. You can only use to perform this type of operation. To export the schema or operational attributes and values, you need to use ldapsearch with the `-L` option as follows:

```
# ldapsearch -L -D "cn=orcladmin" -w "welcome" -h alex -p 389 -b
"cn=subschemaSubentry" -s base "objectclass=*" > alex_oid_schema.ldif
```

---

## Loading a LDIF file created with `ldifwrite`

*by Jeff Hunter, Sr. Database Administrator*

---

### Contents



---

### Overview

The purpose of this document is to provide a script that can be usd to remove operational attributes from a LDIF file before loading it to the directory.

When the OID command tool `ldifwrite` is used to unload directory data, by default, it also unloads operational attributes, such as `orclguid`, `creatorsname`, `createtimestamp`, `modifiersname` and `modifytimestamp`. Then when using the OID command tool `ldapadd` to load this LDIF file, these attributes result in a runtime error: *ldap_add: Constraint violation*.

This procedure will present a script that can be used to remove these entries from the LDIF file allowing you to proceed with the load.

**NOTE:** If the LDIF file was created with `ldapsearch` rather than `ldifwrite`, this is not a problem as operational attributes are not unloaded and written to the LDIF file at all.

## Fix LDIF File For Reloading

1. Copy the following lines into a file named **`fixLDIFFile`** and modify according to your environment. Make sure the ldapmodify command is ALL on a single line.
2. Make the file executable...

```
# chmod +x fixLDIFFile
====================COPY EVERYTHING BELOW THIS
LINE============================
#!/bin/ksh
#
# fix ldifwrite file so it can be loaded into an existing OID
database
#
print "Removing Offending Entries"
egrep -v
"orclguid|creatorsname|modifiersname|createtimestamp|modifytime
stamp|pwdchangedtime" dumped.ldif > new.ldif
print "Removing Offending Entries"
ldapmodify -a -c -p 389 -h alex -D cn=orcladmin -w  -v -f
./new.ldif
print "Retrieve the New Entries"
ldapsearch -p 389 -h alex -b "" -v objectclass=*
====================COPY EVERYTHING UP TO THIS
LINE============================
```

3. Run your command.

   *NOTE: the script will retrieve the entries to verify that the command worked.*

4. See Guide ct to see chapter on Password Policy in OID Admin Guide:

   In addition, the object class top contains these operational attributes, to maintain the user-password state information for each user entry.

   - pwdChangedtime: The timestamp of the user password creation or modification.
   - pwdExpirationWarned: The time at which the first password expiration warning is been sent to the user.
   - pwdFailuretime: The timestamp of consecutive failed login attempts by the user.
   - pwdAccountLockedTime: The time at which the user account was locked.
   - pwdReset: Requirement for the user to change the password, if this attribute is enabled.
   - pwdGraceUseTime: The time stamps of each grace login by the user.

# Installing the Oracle Internet Directory PL/SQL API

*by Jeff Hunter, Sr. Database Administrator*

## Contents



## About the OID PL/SQL API

The PL/SQL API is packaged in the `DBMS_LDAP` package. It is based on the C API described in the previous section.

You can use the Oracle Internet Directory API release 3.0.1 in the following modes:

- SSL-All communication secured using SSL
- Non-SSL-Client-to-server communication not secure

The API uses TCP/IP to connect to an LDAP server. When it does this, it uses, by default, an unencrypted channel. To use the SSL mode, you must use the Oracle SSL call interface. You determine which mode you are using by the presence or absence of the SSL calls in the API usage. You can easily switch between SSL and non-SSL modes.

### Using the PL/SQL API from a Database Trigger

The `DBMS_LDAP` API can be invoked from database triggers to synchronize any changes to a database table with an enterprise-wide LDAP server. The following example illustrates how changes to a table called 'EMP' are synchronized with the data in an LDAP server using triggers for insert, update, and delete.

Examples can be found in the plsql directory under `$ORACLE_HOME/ldap/demo`.

## Building Applications with the OID PL/SQL API

To use the PL/SQL LDAP API, you must first load it into the database. You do this by using a script called `catldap.sql` that is located in the `$ORACLE_HOME/rdbms/admin directory`.

You must be connected as SYSDBA using the SQL*Plus command line tool.

The following is a sample command sequence that you can use to load the DBMS_LDAP package:

```
SQL> CONNECT / AS SYSDBA
SQL>  @?/rdbms/admin/catldap.sql
```

## Dependencies and Limitations

The PL/SQL LDAP API for this release has the following limitations:

- The LDAP session handles obtained from the API are valid only for the duration of the database session. The LDAP session handles cannot be written to a table and re-used in other database sessions.
- Only synchronous versions of LDAP API functions are supported in this release.
- The PL/SQL LDAP API requires a database connection to work. It cannot be used in client-side PL/SQL engines (like Oracle Forms) without a valid database connection.

## PL/SQL Reference

The PL/SQL package `DBMS_LDAP` contains the functions and procedures which can be used by PL/SQL programmers to access data from LDAP servers. This section explains all of the API functions in detail.

### Summary of Subprograms

| Function or Procedure | Description |
|---|---|
| **FUNCTION -- init** | init() initializes a session with an LDAP server. This actually establishes a connection with the LDAP server. |
| **FUNCTION -- simple_bind_s** | The function simple_bind_s can be used to perform simple user name/password based authentication to the directory server. |
| **FUNCTION -- bind_s** | The function bind_s can be used to perform complex authentication to the directory server. |
| **FUNCTION -- unbind_s** | The function unbind_s is used for closing an active LDAP session. |
| **FUNCTION -- compare_s** | The function compare_s can be used to test if a particular attribute in a particular entry has a particular value. |
| **FUNCTION -- search_s** | The function search_s performs a synchronous search in the LDAP server. It returns control to the PL/SQL environment only after all of the search results have been sent by the server or if the search request is 'timed-out' by the server. |
| **FUNCTION -- search_st** | The function search_st performs a synchronous search in the LDAP server with a client side time-out. It returns control to the PL/SQL environment only after all of the search results have been sent by the server or if the search request is 'timed-out' by the client or the server. |

| | |
|---|---|
| **FUNCTION -- first_entry** | The function first_entry is used to retrieve the first entry in the result set returned by either search_s or search_st. |
| **FUNCTION -- next_entry** | The function next_entry() is used to iterate to the next entry in the result set of a search operation. |
| **FUNCTION -- count_entries** | This function is used to count the number of entries in the result set. It can also be used to count the number of entries remaining during a traversal of the result set using a combination of the functions first_entry() and next_entry(). |
| **FUNCTION -- first_attribute** | The function first_attribute() fetches the first attribute of a given entry in the result set. |
| **FUNCTION -- next_attribute** | The function next_attribute() fetches the next attribute of a given entry in the result set. |
| **FUNCTION -- get_dn** | The function get_dn() retrieves the X.500 distinguished name of given entry in the result set. |
| **FUNCTION -- get_values** | The function get_values() can be used to retrieve all of the values associated for a given attribute in a given entry. |
| **FUNCTION -- get_values_len** | The function get_values_len() can be used to retrieve values of attributes that have a 'Binary' syntax. |
| **FUNCTION -- delete_s** | This function can be used to remove a leaf entry in the LDAP Directory Information Tree. |
| **FUNCTION -- modrdn2_s** | The function modrdn2_s() can be used to rename the relative distinguished name of an entry. |
| **FUNCTION -- err2string** | The function err2string() can be used to convert an LDAP error code to string in the local language in which the API is operating. |
| **FUNCTION -- create_mod_array** | The function create_mod_array() allocates memory for array modification entries that will be applied to an entry using the modify_s() functions. |
| **PROCEDURE -- populate_mod_array (String Version)** | Populates one set of attribute information for add or modify operations. This procedure call has to happen after DBMS_LDAP.create_mod_array() is called. |
| **PROCEDURE -- populate_mod_array (Binary Version)** | Populates one set of attribute information for add or modify operations. This procedure call has to happen after DBMS_LDAP.create_mod_array() is called. |
| **FUNCTION -- modify_s** | Performs a synchronous modification of an existing LDAP directory entry. Before calling add_s, we have to call DBMS_LDAP.creat_mod_array () and DBMS_LDAP.populate_mod_array() first. |
| **FUNCTION -- add_s** | Adds a new entry to the LDAP directory synchronously. Before calling add_s, we have to call DBMS_LDAP.creat_mod_array () and DBMS_LDAP.populate_mod_array() first. |
| **PROCEDURE -- free_mod_array** | Frees the memory allocated by DBMS_LDAP.create_mod_array(). |

| | |
|---|---|
| **FUNCTION -- count_values** | Counts the number of values returned by DBMS_LDAP.get_values (). |
| **FUNCTION -- count_values_len** | Counts the number of values returned by DBMS_LDAP.get_values_len (). |
| **FUNCTION -- rename_s** | Renames an LDAP entry synchronously. |
| **FUNCTION -- explode_dn** | Breaks a DN up into its components. |
| **FUNCTION -- open_ssl** | Establishes an SSL (Secure Sockets Layer) connection over an existing LDAP connection. |
| **FUNCTION -- msgfree** | This function frees the chain of messages associated with the message handle returned by synchronous search functions. |
| **FUNCTION -- ber_free** | This function frees the memory associated with a handle to BER ELEMENT. |

## Exception Summary

| Exception Name | Oracle Error Number | Cause of Exception |
|---|---|---|
| **general_error** | 31202 | Raised anytime an error is encountered that does not have a specific PL/SQL exception associated with it. The error string contains the description of the problem in the local language of the user. |
| **init_failed** | 31203 | Raised by DBMS_LDAP.init() if there are some problems. |
| **invalid_session** | 31204 | Raised by all functions and procedures in the DBMS_LDAP package if they are passed an invalid session handle. |
| **invalid_auth_method** | 31205 | Raised by DBMS_LDAP.bind_s() if the authentication method requested is not supported. |
| **invalid_search_scope** | 31206 | Raised by all of the 'search' functions if the scope of the search is invalid. |
| **invalid_search_time_val** | 31207 | Raised by time based search function: DBMS_LDAP.search_st() if it is given an invalid value for the time limit. |
| **invalid_message** | 31208 | Raised by all functions that iterate through a result-set for getting entries from a search operation if the message handle given to them is invalid. |
| **count_entry_error** | 31209 | Raised by DBMS_LDAP.count_entries if it cannot count the entries in a given result set. |
| **get_dn_error** | 31210 | Raised by DBMS_LDAP.get_dn if the DN of the entry it is retrieving is NULL. |
| **invalid_entry_dn** | 31211 | Raised by all the functions that modify/add/rename an entry if they are presented with an invalid entry DN. |
| **invalid_mod_array** | 31212 | Raised by all functions that take a modification array as an argument if they are given an invalid modification array. |
| **invalid_mod_option** | 31213 | Raised by DBMS_LDAP.populate_mod_array if the modification option given is anything other than MOD_ADD, MOD_DELETE or MOD_REPLACE. |
| **invalid_mod_type** | 31214 | Raised by DBMS_LDAP.populate_mod_array if the attribute type that is being modified is NULL. |

| | | |
|---|---|---|
| **invalid_mod_value** | 31215 | Raised by DBMS_LDAP.populate_mod_array if the modification value parameter for a given attribute is NULL. |
| **invalid_rdn** | 31216 | Raised by all functions and procedures that expect a valid RDN if the value of the RDN is NULL. |
| **invalid_newparent** | 31217 | Raised by DBMS_LDAP.rename_s if the newparent of an entry being renamed is NULL. |
| **invalid_deleteoldrdn** | 31218 | Raised by DBMS_LDAP.rename_s if the deleteoldrdn parameter is invalid. |
| **invalid_notypes** | 31219 | Raised by DBMS_LDAP.explode_dn if the notypes parameter is invalid. |
| **invalid_ssl_wallet_loc** | 31220 | Raised by DBMS_LDAP.open_ssl if the wallet location is NULL but the SSL authentication mode requires a valid wallet. |
| **invalid_ssl_wallet_password** | 31221 | Raised by DBMS_LDAP.open_ssl if the wallet password given is NULL. |
| **invalid_ssl_auth_mode** | 31222 | Raised by DBMS_LDAP.open_ssl if the SSL authentication mode is not one of 1, 2 or 3. |

**Data-Type Summary**

| Data-Type | Purpose |
|---|---|
| **SESSION** | Used to hold the handle of the LDAP session. Nearly all of the functions in the API require a valid LDAP session to work. |
| **MESSAGE** | Used to hold a handle to the message retrieved from the result set. This is used by all functions that work with entries attributes and values. |
| **MOD_ARRAY** | Used to hold a handle into the array of modifications being passed into either modify_s() or add_s(). |
| **TIMEVAL** | Used to pass time limit information to the LDAP API functions that require a time limit. |
| **BER_ELEMENT** | Used to hold a handle to a BER structure used for decoding incoming messages. |
| **STRING_COLLECTION** | Used to hold a list of VARCHAR2 strings which can be passed on to the LDAP server. |
| **BINVAL_COLLECTION** | Used to hold a list of RAW data which represent binary data. |
| **BERVAL_COLLECTION** | Used to hold a list of BERVAL values that are used forpopulating a modification array. |

---

## Oracle Internet Directory PL/SQL API Examples

*by Jeff Hunter, Sr. Database Administrator*

---

### LDAP_SEARCH_EXAMPLE_PROC (OID PL/SQL API Example)

## PL/SQL Source Code

```
CREATE OR REPLACE PROCEDURE LDAP_SEARCH_EXAMPLE_PROC AS

    ldap_host       VARCHAR2(512);          -- The LDAP Directory Host
    ldap_port       VARCHAR2(512);          -- The LDAP Directory Port
    ldap_user       VARCHAR2(512);          -- The LDAP Directory User
    ldap_passwd     VARCHAR2(512);          -- The LDAP Directory Password
    ldap_baseDN     VARCHAR2(512);          -- The starting (base) DN
    retval          PLS_INTEGER;            -- Used for all API return
values.
    my_session      DBMS_LDAP.SESSION;      -- Used to store our LDAP
Session
    res_attrs       DBMS_LDAP.STRING_COLLECTION;    -- A String Collection
used
                                                    --   to specify which
                                                    --   attributes to
return
                                                    --   from the search.
    search_filter   VARCHAR2(512);          -- A simple character string
used to
                                            --   store the filter
(criteria) for
                                            --   the search.
    res_message     DBMS_LDAP.MESSAGE;      -- Used to store the message
                                            --   (results) of the search.
    temp_entry      DBMS_LDAP.MESSAGE;      -- Used to store entries
retrieved
                                            --   from the LDAP search to
print
                                            --   out at a later time.
    entry_index     PLS_INTEGER;            -- Used as a counter while
looping
                                            --   through each entry. As we
                                            --   retrieve an entry from the
LDAP
                                            --   directory, we increase the
                                            --   counter by one.
    temp_dn         VARCHAR2(512);          -- After each entry is
retrieved
                                            --   from the LDAP directory
(from
                                            --   the search), we want to
use
                                            --   this variable to extract,
store
                                            --   and print out the DN for
each
                                            --   entry.
    temp_attr_name  VARCHAR2(512);          -- After retrieving an entry
from
                                            --   LDAP directory, we will
want to
                                            --   walk through all of the
                                            --   returned attributes. This
                                            --   variable will be used to
store
                                            --   each attribute name as we
loop
                                            --   through them.
```

```
    temp_ber_elmt   DBMS_LDAP.BER_ELEMENT;
    attr_index      PLS_INTEGER;            -- Used as a counter variable
for
                                            --   each entry returned for
each
                                            --   entry.
    temp_vals       DBMS_LDAP.STRING_COLLECTION;    -- Used to extract,
store,
                                               --   and print each of
the
                                               --   values from each
                                               --   attribute.

BEGIN

    DBMS_OUTPUT.ENABLE(1000000);

    retval := -1;

    -- ------------------------------------------------------------------
-----
    -- Customize the following variables as needed.
    -- ------------------------------------------------------------------
-----
    ldap_host    := 'alex' ;
    ldap_port    := '389';
    ldap_user    := 'cn=orcladmin';
    ldap_passwd  := 'welcome';
    ldap_baseDN  := 'o=airius.com';

    -- ------------------------------------------------------------------
-----
    -- Print out variables.
    -- ------------------------------------------------------------------
-----
    DBMS_OUTPUT.PUT_LINE('DBMS_LDAP Search Example');
    DBMS_OUTPUT.PUT_LINE('-----------------------------------------------
--');
    DBMS_OUTPUT.PUT_LINE(RPAD('LDAP Host ', 25, ' ') || ': ' || ldap_host);
    DBMS_OUTPUT.PUT_LINE(RPAD('LDAP Port ', 25, ' ') || ': ' || ldap_port);
    DBMS_OUTPUT.PUT_LINE(RPAD('LDAP User ', 25, ' ') || ': ' || ldap_user);
    DBMS_OUTPUT.PUT_LINE(RPAD('LDAP Base ', 25, ' ') || ': ' ||
ldap_baseDN);

    -- ------------------------------------------------------------------
-----
    -- We want all exceptions from DBMS_LDAP to be raised.
    -- ------------------------------------------------------------------
-----
    DBMS_LDAP.USE_EXCEPTION := TRUE;

    -- ------------------------------------------------------------------
-----
    -- Obtain an LDAP session. The init() function initializes a session
with an
    -- LDAP server. This actually establishes a connection with the LDAP
server
    -- and returns a handle to the session which can be used for further
    -- calls into the API.
    -- ------------------------------------------------------------------
-----
```

```
    my_session := DBMS_LDAP.INIT(ldap_host, ldap_port);

    DBMS_OUTPUT.PUT_LINE (
        RPAD('LDAP Session ', 25, ' ') || ': ' ||
        RAWTOHEX(SUBSTR(my_session, 1, 16)) ||
        ' - (returned from init)'
    );

    -- ------------------------------------------------------------------
-----
    -- Bind to the directory. The function simple_bind_s can be used to
perform
    -- simple username/password based authentication to the directory
server.
    -- The username is a directory distinguished name. This function can be
    -- called only after a valid LDAP session handle is obtained from a
call to
    -- DBMS_LDAP.init(). If the connection was successful, it will return:
    -- DBMS_LDAP.SUCCESS. This function can raise the following exceptions:
    --      invalid_session : Raised if the session handle ld is invalid.
    --      general_error   : For all other errors. The error string
associated
    --                        with this exception will explain the error in
    --                        detail.
    -- ------------------------------------------------------------------
-----
    retval := DBMS_LDAP.SIMPLE_BIND_S(my_session, ldap_user, ldap_passwd);

    DBMS_OUTPUT.PUT_LINE(
        RPAD('simple_bind_s Returned ', 25, ' ') || ': '|| TO_CHAR(retval)
    );

    -- ------------------------------------------------------------------
-----
    -- Before actually performing the sort, I want to setup the attributes
I
    -- would like returned. To do this, I declared a "String Collection"
that
    -- will be used to store all of the attributes I would like returned.
    --
    --      If I wanted to return all attributes, I would specify:
    --          res_attrs(1) := '*';
    --
    --      If I wanted multiple (specified) attributes, I would specify:
    --          res_attrs(1) := 'cn';
    --          res_attrs(2) := 'telephoneNumber';
    -- ------------------------------------------------------------------
-----
    res_attrs(1) := 'uid';
    res_attrs(2) := 'cn';
    res_attrs(3) := 'telephoneNumber';

    -- ------------------------------------------------------------------
-----
    -- Finally, before performing the actual search, I want to specify the
    -- criteria I want to search on. This will be passed as the "filter"
    -- parameter to the actual search.
    --
    --      If you wanted all of the entries in the directory to be
returned,
    --      you could simply specify:
```

```
    --          search_filter   := 'objectclass=*';
    --
    --      You could also refine your search my specify a criteria like
the
    --      following:
    --          search_filter   := 'cn=Jeff*';
    -- ------------------------------------------------------------------
-----
    search_filter   := 'cn=Jeff*';

    -- ------------------------------------------------------------------
-----
    -- Finally, let's issue the search. The function search_s performs a
    -- synchronous search in the LDAP server. It returns control to the
PL/SQL
    -- environment only after all of the search results have been sent by
the
    -- server or if the search request is 'timed-out' by the server.
    --
    -- Let's first explain some of the incoming parameters:
    --      ld       : A valid LDAP session handle.
    --      base     : The dn of the entry at which to start the search.
    --      scope    : One of SCOPE_BASE     (0x00)
    --                        SCOPE_ONELEVEL (0x01)
    --                        SCOPE_SUBTREE  (0x02)
    --                 indicating the scope of the search.
    --      filter   : A character string representing the search filter.
The
    --                 value NULL can be passed to indicate that the filter
    --                 "(objectclass=*)" which matches all entries is to be
    --                 used.
    --      attrs    : A collection of strings indicating which attributes
to
    --                 return for each matching entry. Passing NULL for
this
    --                 parameter causes all available user attributes to be
    --                 retrieved. The special constant string NO_ATTRS
("1.1")
    --                 MAY be used as the only string in the array to
indicate
    --                 that no attribute types are to be returned by the
server.
    --                 The special constant string ALL_USER_ATTRS ("*") can
be
    --                 used in the attrs array along with the names of some
    --                 operational attributes to indicate that all user
    --                 attributes plus the listed operational attributes
are to
    --                 be returned.
    --      attronly : A boolean value that MUST be zero if both attribute
types
    --                 and values are to be returned, and non-zero if only
types
    --                 are wanted.
    --      res      : This is a result parameter which will contain the
results
    --                 of the search upon completion of the call. If no
results
    --                 are returned, res is set to NULL.
    --
    -- Now let's look at the two output parameters:
```

```
    --      PLS_INTEGER
    --      (function return)   : DBMS_LDAP.SUCCESS if the search operation
    --                            succeeded. An exception is raised in all
other
    --                            cases.
    --      res (OUT parameter) : If the search succeeded and there are
entries,
    --                            this parameter is set to a NON-NULL value
    --                            which can be used to iterate through the
    --                            result set.
    -- ------------------------------------------------------------------
-----
    retval := DBMS_LDAP.SEARCH_S(
          ld         =>  my_session
        , base       =>  ldap_baseDN
        , scope      =>  DBMS_LDAP.SCOPE_SUBTREE
        , filter     =>  search_filter
        , attrs      =>  res_attrs
        , attronly   =>  0
        , res        =>  res_message
    );

    DBMS_OUTPUT.PUT_LINE(
        RPAD('search_s Returned ', 25, ' ') || ': ' || TO_CHAR(retval)
    );
    DBMS_OUTPUT.PUT_LINE (
        RPAD('LDAP Message ', 25, ' ') || ': ' ||
        RAWTOHEX(SUBSTR(res_message, 1, 16)) ||
        ' - (returned from search_s)'
    );

    -- ------------------------------------------------------------------
-----
    -- After the search is performed, the API stores the count of the
number of
    -- entries returned.
    -- ------------------------------------------------------------------
-----
    retval := DBMS_LDAP.COUNT_ENTRIES(my_session, res_message);
    DBMS_OUTPUT.PUT_LINE(
        RPAD('Number of Entries ', 25, ' ') || ': ' || TO_CHAR(retval)
    );
    DBMS_OUTPUT.PUT_LINE('-----------------------------------------------
--');


    -- ------------------------------------------------------------------
-----
    -- Retrieve the first entry.
    -- ------------------------------------------------------------------
-----
    temp_entry := DBMS_LDAP.FIRST_ENTRY(my_session, res_message);
    entry_index := 1;

    -- ------------------------------------------------------------------
-----
    -- Loop through each of the entries one by one.
    -- ------------------------------------------------------------------
-----
    WHILE temp_entry IS NOT NULL LOOP
```

```
        -- ------------------------------------------------------------------
-----
        -- Print out the current entry.
        -- ------------------------------------------------------------------
-----
        temp_dn := DBMS_LDAP.GET_DN(my_session, temp_entry);
        DBMS_OUTPUT.PUT_LINE (' dn: ' || temp_dn);

        temp_attr_name := DBMS_LDAP.FIRST_ATTRIBUTE(
              my_session
            , temp_entry
            , temp_ber_elmt
        );
        attr_index := 1;
        WHILE temp_attr_name IS NOT NULL LOOP

            temp_vals := DBMS_LDAP.GET_VALUES(my_session, temp_entry,
temp_attr_name);
            IF temp_vals.COUNT > 0 THEN
                FOR i IN temp_vals.FIRST..temp_vals.LAST LOOP
                    DBMS_OUTPUT.PUT_LINE(
                        RPAD('   ' || temp_attr_name, 19, ' ') ||
                        ': ' || SUBSTR(temp_vals(i), 1, 200)
                    );
                END LOOP;
            END IF;
            temp_attr_name := DBMS_LDAP.NEXT_ATTRIBUTE(   my_session
                                                        , temp_entry
                                                        , temp_ber_elmt);
            attr_index := attr_index + 1;

        END LOOP;

        temp_entry := DBMS_LDAP.NEXT_ENTRY(my_session, temp_entry);

DBMS_OUTPUT.PUT_LINE('==================================================')
;
        entry_index := entry_index + 1;
    END LOOP;

    -- ------------------------------------------------------------------
-----
    -- Unbind from the directory
    -- ------------------------------------------------------------------
-----
    retval := DBMS_LDAP.UNBIND_S(my_session);
    DBMS_OUTPUT.PUT_LINE(RPAD(
        'unbind_res Returned ', 25, ' ') || ': ' ||
        TO_CHAR(retval)
    );

    -- ------------------------------------------------------------------
-----
    -- Handle Exceptions
    -- ------------------------------------------------------------------
-----
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE(' Error code : ' || TO_CHAR(SQLCODE));
            DBMS_OUTPUT.PUT_LINE(' Error Message : ' || SQLERRM);
            DBMS_OUTPUT.PUT_LINE(' Exception encountered .. exiting');
```

```
END;
/
```

## Results

```
SQL> set serveroutput on
SQL> call scott.ldap_search_example_proc();

DBMS_LDAP Search Example
------------------------------------------------
LDAP Host                 : alex
LDAP Port                 : 389
LDAP User                 : cn=orcladmin
LDAP Base                 : o=airius.com
LDAP Session              : FC88B8040003100E - (returned from init)
simple_bind_s Returned    : 0
search_s Returned         : 0
LDAP Message              : FC88AFC8FC88F154 - (returned from search_s)
Number of Entries         : 4
------------------------------------------------
 dn: uid=jcampai2, ou=People, o=airius.com
   uid             : jcampai2
   cn              : Jeffrey Campaigne
   telephonenumber : +1 408 555 7393
================================================
 dn: uid=jmuffly, ou=People, o=airius.com
   uid             : jmuffly
   cn              : Jeff Muffly
   telephonenumber : +1 408 555 5287
================================================
 dn: uid=jvaughan, ou=People, o=airius.com
   uid             : jvaughan
   cn              : Jeff Vaughan
   telephonenumber : +1 408 555 4543
================================================
 dn: uid=jvedder, ou=People, o=airius.com
   uid             : jvedder
   cn              : Jeff Vedder
   telephonenumber : +1 408 555 4668
================================================
unbind_res Returned       : 0
```