

*The Technical DBA Hints and Techniques  
For Oracle Apps  
Oracle Application Users Group  
Craig Ward  
Southern Systems Solutions*

---

## **Introduction**

---

### ***Database and APPS Administration***

The technical portion of Oracle Applications system consists of two basic parts. The Relational Data Base Management System (RDBMS) and the Applications. The RDBMS is the Oracle engine that stores and retrieves the data. The Applications (Apps) are a set of programs that process and manipulate the data and then give it to the database to store. Both of these areas require a qualified technical administrator (DBA). As there are many books and classes on Database Administration, this text will concentrate on the Apps administration.

The Oracle Applications require two types of system administration. The first is the functional administration, the second is technical administration. The functional administration consists of setting up Apps accounts, privileges, printers etc. Oracle offers a class and it is usually performed by one of the functional users. The technical administration consists of installations, upgrades, and other types of maintenance. There are no classes on this type of administration, yet it must be done properly in order for the Oracle Apps to operate correctly. This text addresses the technical administration almost exclusively.

---

### ***Apps Overview***

The Oracle Apps are set up with a system I refer to as the TOP system. It starts with APPL\_TOP which is an environment variable that points to the head of the apps executables directory system. Each major section of the Apps have a major sub-directory under APPL\_TOP. Each section is also defined as a TOP directory i.e. the head or sub-directory defined by AR\_TOP and it goes to GL\_TOP, MFG\_TOP etc. until all the installed sub-systems, either full or shared, are defined. This method allows the Oracle Apps to be installed on a system without 1.5G of space on one disk volume. It also allows for more

than one installation of the Apps on the same system without fear of overlap.

The Oracle Apps also make use of all the features of the Oracle RDBMS including views, packages, and stored procedures. Correctly installing and maintaining these objects is a major part of the technical DBA's job. This is especially true with the smart client version of the Apps (10SC). This paper will give detailed guidelines on maintaining these objects.

---

## **System Setup**

---

### ***The Optimal Flexible Architecture (OFA)***

#### **Definition**

The Optimal Flexible Architecture<sup>1</sup> is a method for laying out the database and the applications so that multiple versions of the oracle or applications software can exist on the same computer without affecting each other. It also allows upgrades to be performed without damaging the stable production environment.

The OFA setup is straight forward and simple. Following it's guidelines makes life easier for both the short and long term.

#### **Oracle Setup**

The Oracle OFA has four basic parts. The ORACLE\_BASE, ORACLE\_HOME, admin, and oradata, as well as some minor parts like local, aps7 etc. All parts except oradata hang off ORACLE\_BASE. All sub-directories can be on the same mount point or use symbolic links to make it appear as one structure.

All the version specific software & files reside under product/ORACLE\_HOME. All the instance specific files reside under admin. The admin allows each instance to have it's own space. Multiple ORACLE\_HOME's can reside under the ORACLE\_BASE/product directory. This is either to

---

<sup>1</sup> See Oracle Publication A19308-1 [The OFA Standard](#) by Cary Millsap for details concerning the OFA standard.

separate for upgrades and patches or for different version.

The oradata directories reside under the mount points of the OS and contain the Oracle data files. There typically are many mount point as Oracle works best when it is spread out on separate physical disks. As a rule a mount point should be at least one physical disk. If it is more it should increment a full disk at a time. Under each oradata directory is a directory for each instance that has files on the mount point. This allows multiple instances of Oracle to have data on the same mount point without mixing the data files.

## **Apps Setup**

The apps are setup under a main directory called APPL\_TOP. Under the APPL\_TOP directory a sub-directory exists for each module required. I.e. ar, gl, inv etc. Under each of the product directories there is a version sub-directory. (\$APPL\_TOP/ar/7.0.152) This allows one or more versions of AR to exist on the same system without problems. Each of the product directories also have an environment variable set to them. AR\_TOP = /u01/applmgr/apps/ar/7.0.152. This allows systems to be configured that do not have enough disk space to allow a complete application of the apps software on one disk.<sup>2</sup>

---

## **Multiple Environments**

Using a combination of environment variables and symbolic links Oracle and the apps can have multiple installations on the same physical machine and never have any cross over between installations. This allows for a production instance and test instance to co-exist on the same system without corruption. All patches and upgrades should be tested on a test or dba instance prior to being applied to the production instance. This is true for both Oracle RDBMS software and the Oracle Apps software. Any production Oracle Apps system should have at least two ORACLE\_HOME's and two APPL\_TOP's.

---

<sup>2</sup> If your system allows symbolic links then it is best to use the links to make it appear as everything is installed under one mount point.

---

# **Oracle Apps Administration**

---

## **Initial Installation**

The Oracle installation/upgrade manuals are usually very good. They give both high and low level descriptions of what needs to be done. The manual also gives details about pre-work that is required for a successful installation. The initial installation uses an Oracle tool called adaimgr. It allows the sizing of extents, defining where to put data and indexes, and limit the installation to only the modules purchased. This tool does not build the database or name the tablespaces and data files. The database must be sized and built prior to using the adaimgr.

## **Full, Shared, or Non Installation**

In the adaimgr it requires the dba to indicate which modules to install and what type of installation is required. The options are full, shared and non-installed. A full install is required if the module has been purchased and is going to be implemented. A Shared install is required if one of the modules being installed requires some of the tables of another module. For Example; The purchasing module requires the table structure from the Human Resources module. The purchasing module would be a full install and adaimgr would automatically flag the Human Resources module for a shared install. The only items that have to be flagged are the full installs. Any required shared installs will be automatically flagged.

## **Data Base Configuration**

Each module that is fully installed should have its own tablespace for data and a separate one for indexes. A module receiving a shared install may have its own tablespaces or they may be combined into a shared tablespace for data and a shared tablespace for indexes. There is not a hard rule to determine which way this should be done, but if there is a chance the module may be implemented in the future it is a good idea to create the tablespaces for the module. Put all the other shared modules in the shared tablespaces. The installation manuals provide information that is helpful to physically sizing the different tablespaces for each module.

## Sizing Percent

The sizing percent addresses the way the data will be built in the tablespaces. The 100% contains extent sizes large enough to handle an average installation for a year. This will be different for each system but a 100% sizing will ensure the apps implementation will work. The most recent books indicate that shared modules should also be sized at 100%. The reason for this is to make it easier to convert from a shared install to a full install. All tables and indexes are installed on a shared install so if the sizing is 100% the conversion is easy. If however, there is no chance the system will be converting a particular module to full, the sizing percentage may be reduced. A good idea is to configure any shared module that was given it's own tablespaces at 100%. The ones in the shared tablespaces should be configured at a much smaller percentage.

---

## Patches

### Patch Overview

The oracle apps are fixed by patches. These patches come in client side and server side variety. The patches generally come out from oracle in a compressed tar format. This format is used whether you download the patch off the external ftp site, oracle-ftp.us.oracle.com/outgoing, or receive it on some other media.

The server side patches should be put in \$APPL\_TOP/patch directory. The name will be similar to 12345.1070t.Z. The 12345 is the bug number. The 1070 is the version of the apps. The t means tar. The Z means it is compressed. The patch must be decompressed first with the UNIX command `uncompress` in the syntax of `uncompress 12345.1070t` without the Z. `uncompress` will error if the file has a small z or if the Z is included in the filename. Once the file is decompressed the .Z is eliminated. After the decompression the files must be extracted from the tar file as follows: `tar xvf 12345.1070`. This command will use the file 12345.1070 rather than the tape drive. It will also create any directory structure required by the patch. Once the patch is decompressed and removed from the tar archive it can be applied.

A patch consists of a patch.driv, possibly other 12345.driv files, and all the files in appropriate sub-directories required to apply the patch. The patch will contain the driver to solve the bug identified to

that patch and any other patches that it requires in order to work. If a required patch is not included as part of the current patch the readme will state to apply patch 12345 prior to applying the current patch.

### Apply The Patch

Applying a patch to the Oracle Apps requires Oracles Utility `adpatch`. Specific instructions for each patch are in a readme file that comes with the patch. After extracting the patch.tar file the readme file is in the patch number directory. To apply the patch do the following:

1. Untar the patch file `"tar xvf 12345.1070.t"`. (This creates a sub-directory of the patch number, 12335, and all the sub-directories needed to apply the patch.)
2. `cd 12345`
3. Read the readme file and take note of all special instructions as these are the installation instructions. If there is more than one .drv file the readme will tell whether or not to apply them also and in what order. The special instructions will say after applying the patch, apply `db12345.driv3`. Take note of what driv's are to be applied and apply them in the order they appear in the readme instructions<sup>4</sup>.
4. Run Oracle's `adpatch` utility and answer the questions appropriately. Always run this from the directory that contains the .drv files. Always run the patch.driv file first unless specifically directed by the readme.
5. After the patch has applied check the `adpatch.log` for errors that may not have been caught by `adpatch`. The logfiles are in the \$APPL\_TOP/install/log directory. The main log is `adpatch.log`. If any programs are relinked there will be an `adrelink.log` file. These files are accumulative so it may be appropriate to remove them prior to applying a new patch. If there are no errors continue. If there is an error see the section of this document "OOP's it doesn't work".
6. Rerun steps 4 & 5 as many times as you have .drv files to apply.
7. Perform any other instructions that may be given in the special instructions. (This may require involvement from the functional team.)
8. Check to see if all the objects are compiled. (Use `chk_objs.sql` from Appendix A.) If there

---

<sup>3</sup> 12345 is replaced by the particular patch number that is being referenced by the patch

<sup>4</sup> The order may not be critical but it is safe that way.

are objects that are not compiled they must be compiled before the apps will work correctly.

## Errors & Logs

All the Oracle tools write everything that goes to the screen and all errors to a logfile. These logs are accumulative and can become very large. Periodically they must be purged. Typically once a patch has been applied error free the log is uninteresting and can be purged.

The different Oracle tools each have their own patch log. i.e. adpatch creates adpatch.log, adrelink creates adrelink.log, adadmin creates adadmin.log etc. These logs record everything that is output to standard output or standard error.<sup>5</sup>

## OOP's It Doesn't Work

Most of the patches applied to a well maintained implementation will apply without any problems. However, with any system as large and complex as the Oracle Apps and with as many different site specific considerations, there is always a chance a patch will have a problem applying. The adpatch.log or one of the other logs that are created in the \$APPL\_TOP/install/log directory is the first place to look in solving a problem with a patch.

Different parts of the patch install write to different logs. To determine which log may contain details to the problem, cd to the log directory and type "ls -lt | more". This will show the most recent written logs and the top ones will usually hold the error messages needed for evaluation. If the answer is not obvious, these errors often require a call to Oracle Support<sup>6</sup>.

## Upgrades

There are many pieces to upgrade in the Oracle Apps. The RDBMS, Oracle Apps, Smart Client (GUI) server and Smart Client client all periodically need upgraded and patched. Different tools and procedures are required for each piece. Oracle usually provides very good and detailed documentation to apply each of the different patches and upgrades<sup>7</sup>. However one rule applies to all upgrades and patches regardless of which piece of the puzzle is being patched or upgraded. NEVER NEVER apply a patch or upgrade to a production

instance without first applying it to a completely independent test instance and testing it and having the users test it thoroughly. This by definition requires all patches and upgrades to be performed at least twice. The extra effort and cost will be worth every minute and penny the first time a bad patch or upgrade is applied.

The RDBMS patches and upgrades are rare but are required from time to time. The most common upgrade is the 10SC for systems using the Smart Client GUI. The 10SC is issued at Prod levels. The most current being Prod 16.1. With each prod release Oracle has fixed a host of bugs and the 10SC system works much better. The 10SC prod upgrades require a server side upgrade that is applied much like patches using adpatch and a client side upgrade that is handled like a new install on the work station. Prior to starting an upgrade ensure all the objects in the system are valid and all patches are completed. Once this is done follow the instruction book sent out by Oracle.

---

# Oracle Apps Implementation

## Overview

An Oracle Apps implementation requires two teams. The functional and the technical. The functional team set the Apps up from a user configuration point of view. They map the business to the Apps. The technical team sets up the databases, installs and maintains the applications using the database not the applications. There is a functional system administrator that is responsible for setting up users and responsibilities and other day to day activities. This person with the help of the technical dba will also set up printers and perform other house cleaning tasks. The functional system administrator is not a dba, but is usually a user that enjoys working with some of the more technical aspects of the application. This person often reports to the finance department or one of the other user groups rather than the MIS dept.

## The Technical Layout

An Oracle Apps implementation requires multiple instances of Oracle for the various requirements of the functional team. Some functional teams will require more than others. The more common instances required are used as follows:

- The MT instance is kept up to date with approved setups, patches and upgrades but is not corrupted with experimental data. (MT stands

---

<sup>5</sup> Standard output is all data that is sent to the screen. Standard error is not always sent to the screen and this is why the logs should always be checked in addition to watching the screen.

<sup>6</sup> For suggestions to troubleshoot the more common problems See Appendix A.

<sup>7</sup> There are exceptions to this. The Apps 10.6 to 10.6.1 instructions were very poor. This is the exception not the rule.

for empty) A copy of this instance is used at checkpoints throughout the development cycle to refresh the other instances. It is imperative that no setups be put into this instance until they are approved and signed off by the end client.

- The dev instance is where the functional team will perform it's work. This instance usually is corrupted with experimental data and setups during the development process and will probably need refreshed during the development life cycle. This instance is used also to refresh some of the others at times in the life cycle.
- The Conference Room Pilot (crp) instance is used to allow the user community to work through all of the applications and approve them. This starts with a copy of MT or if dev is clean enough it can be used. After the pilot is over this instance can go away or be used for some other purpose.
- The dba instance is used to test all patches and upgrades on. It must have it's own ORACLE\_HOME and APPL\_TOP. This database can be refreshed by dev when needed to test a patch or other reason at the discretion of the dba. This should be limited in it's development use as it may be refreshed at any time and data lost.

There may be other instances required at various times during the development cycle. The limitation of disk space and machine resources are the only limitations.

## Copy Database A To Database B

There are many ways to get data from one database to another. They range from building a database and importing data from another database to selecting and importing a subset of the data to physically copying a database onto another, renaming it and rebuilding the control file. Copying the database is the easiest and quickest way to duplicate the database. The downside is the copy takes exactly as much space as the source database and care must be taken not to overwrite or lose part of the source database. The speed of this method far outweighs the downside. To copy the database use the following steps:

1. Determine where all the pieces of the new database will be put.
2. Take down the source database.

3. Back up the source database
4. Make sure the ORACLE\_HOME and APPL\_TOP the new database will use are created.
5. Edit the /etc/oratab and add the new instance with it's proper ORACLE\_HOME.
6. If it is necessary to create a new ORACLE\_HOME and/or APPL\_TOP it can be done as follows:
  - cd \$ORACLE\_HOME
  - cp -rp \* NEW\_ORACLE\_HOME (Same for APPL\_TOP)
7. Use the following cpdb script<sup>8</sup> to create a template to use to perform the copy.

```
#!/bin/sh
#
*****
*****
# Filename   : cpdb - Version 1.1
# Author    : Craig A. Shallahamer
#           : John Strother
# Original   : 16-JAN-95
# Last Update : 2/4/95 - 14:23:07
# Description : Create a rough script to copy a given database.
# Usage      : $ cpdb <old db> <new db> <redo dir> <cr script>
#
*****
*****

if [ $# -lt 4 ]
then
  echo "Usage: $0 <old db> <new db> <redo dir> <cr script>"
  echo "Example: $0 prod test u01 crtest.sql"
  exit 1
else
  olddb=$1
  newdb=$2
  dir1=$3 # where new control files and redo log files go (E.g.,
/u01)
  of=$4 # output file
  #echo "1=$1 2=$2 3=$3 4=$4"
fi

rm -f $of
touch $of

echo "#!/bin/sh" >> $of
echo "#
*****
*****" >> $of
echo "# Filename   : cpdb - Version 1.1" >> $of
echo "# Author    : Craig A. Shallahamer" >> $of
```

<sup>8</sup> This script was written by Craig Shallahamer of Oracle Corporation. It is a part of the APS7 suite of tools. This script assumes an OFA compliant database structure.

```

echo "#          John Strother" >> $of
echo "# Original   : 16-JAN-95" >> $of
echo "# Last Update : 2/4/95 - 14:23:07" >> $of
echo "# Description : Base db copy script. Created from
crdb.sh" >> $of
echo "# Usage      : ${of}" >> $of
echo "#
*****
**" >> $of
echo ""

echo "#" >> $of
echo "# Take out the below 'exit' to enable this script." >> $of
echo "#" >> $of
echo "echo \"Exiting $0 because modifications not complete.\"\"
>> $of
echo "exit 1" >> $of
echo "" >> $of

echo "#" >> $of
echo "# *** Shutdown 'old' database system" >> $of
echo "#" >> $of
echo "ORACLE_SID=${olddb}" >> $of
echo "svrmgrl <<EOF" >> $of
echo "connect internal" >> $of
echo "shutdown immediate" >> $of
echo "exit" >> $of
echo "EOF" >> $of

#
# Create file copy template
#

echo "#" >> $of
echo "# *** Copy database files" >> $of
echo "# Modify the 'to' file name. Especially watch the
SID/DBNAME" >> $of
echo "#" >> $of
ls -l /*/oradata/${olddb}/* | grep -v ".log" | \
grep -v ".ctl" | awk '{print "cp " $1 " " $1}' >> $of

#
# Create "create control" file template
#
echo "#" >> $of
echo "# *** Create control file" >> $of
echo "# Modifications:" >> $of
echo "# 1. Replace database file 'quotes' from '<' and '>' to '\\\".
>> $of
echo "# 2. Remove the last reuse log comma." >> $of
echo "# 3. Change the <<${olddb}>> in the database file name
to \
<<${newdb}>>" >> $of
echo "#" >> $of
echo "" >> $of
echo "" >> $of
echo "ORACLE_SID=${newdb};export ORACLE_SID" >> $of
echo "svrmgrl lmode=y <<EOF" >> $of
echo "connect internal" >> $of
echo "shutdown immediate" >> $of
echo "startup nomount" >> $of
echo "create controlfile set database ${newdb}" >> $of
echo "logfile '${dir1}/oradata/${newdb}/redo01a.log' size 2m," >>
$of
echo " '${dir1}/oradata/${newdb}/redo02a.log' size 2m" >>
$of
echo " resetlogs" >> $of
echo "datafile" >> $of

```

```

ls -l /*/oradata/${olddb}/* | grep -v ".log" |
grep -v ".ctl" | awk '{print "<" $1 "> reuse,}"' >> $of
echo "maxdatafiles 1022" >> $of
echo "maxlogfiles 32" >> $of
echo "maxlogmembers 5" >> $of
echo "maxinstances 1" >> $of
echo "/" >> $of
echo "alter database open resetlogs;" >> $of
echo "EOF" >> $of

#
# Create (actually copy the old) new init.ora and config.ora file
#
for i in pfile udump adump cdump bdump exp arch logbook
create;
do
echo "Making admin ${i} directory..."
mkdir -p ${ORACLE_BASE}/admin/${newdb}/${i}
done

cp -i ${ORACLE_BASE}/admin/${olddb}/pfile/init${olddb}.ora
\
${ORACLE_BASE}/admin/${newdb}/pfile/init${newdb}.ora

cp -i ${ORACLE_BASE}/admin/${olddb}/pfile/config.ora \
${ORACLE_BASE}/admin/${newdb}/pfile/config.ora

echo
*****
*****"
echo ""
echo "The new init.ora and config.ora files have been created.
Modify them"
echo "them before ${of} is run."
echo ""
echo "The files are:"
echo " 1.
${ORACLE_BASE}/admin/${newdb}/pfile/init${newdb}.ora"
echo " 2. ${ORACLE_BASE}/admin/${newdb}/pfile/config.ora"
echo ""
echo ""
echo "The copy database script ${of} has been created which
contains the rough"
echo "commands to actually perform the database copy."
echo ""
echo "You MUST modify the script before running it."
echo ""
echo "Good Luck and don't call me!"
echo ""
echo
*****
*****"

8. This script produces the following output that
will allow the copying of the database and a new
control file. Run the script by typing "cpdb
src_db new_db redo_dir shell" ex. cpdb dev dba
/u01 crdba.sh

#!/bin/sh
#
*****
**
# Filename : cpdb - Version 1.1
# Author : Craig A. Shallahamer

```

```

#           John Strother
# Original   : 16-JAN-95
# Last Update : 2/4/95 - 14:23:07
# Description : Base db copy script. Created from crdb.sh
# Usage      : crdba.sh
#
*****
**
#
# Take out the below 'exit' to enable this script.
#
echo "Exiting /home/oracle/local/aps7/bin/cpdb because
modifications not complet
e."
exit 1

#
# *** Shutdown 'old' database system
#
ORACLE_SID=apps
svrmgrl <<EOF
connect internal
shutdown immediate
exit
EOF
#
# *** Copy database files
# Modify the 'to' file name. Especially watch the SID/DBNAME
#
cp /u04/oradata/apps/crpd02.dbf /u04/oradata/apps/crpd02.dbf
cp /u04/oradata/apps/ecd01.dbf /u04/oradata/apps/ecd01.dbf
cp /u04/oradata/apps/engd01.dbf /u04/oradata/apps/engd01.dbf
cp /u04/oradata/apps/fad01.dbf /u04/oradata/apps/fad01.dbf
cp /u04/oradata/apps/gld01.dbf /u04/oradata/apps/gld01.dbf
cp /u04/oradata/apps/gld02.dbf /u04/oradata/apps/gld02.dbf
cp /u04/oradata/apps/interim01.dbf
/u04/oradata/apps/interim01.dbf
cp /u04/oradata/apps/invx01.dbf /u04/oradata/apps/invx01.dbf
cp /u04/oradata/apps/invx02.dbf /u04/oradata/apps/invx02.dbf
cp /u04/oradata/apps/mrpx01.dbf /u04/oradata/apps/mrpx01.dbf
cp /u04/oradata/apps/mrpx02.dbf /u04/oradata/apps/mrpx02.dbf
cp /u04/oradata/apps/pox01.dbf /u04/oradata/apps/pox01.dbf
cp /u04/oradata/apps/rgx01.dbf /u04/oradata/apps/rgx01.dbf
cp /u04/oradata/apps/shd01.dbf /u04/oradata/apps/shd01.dbf
cp /u13/oradata/apps/rbs01.dbf /u13/oradata/apps/rbs01.dbf
cp /u13/oradata/apps/rbs02.dbf /u13/oradata/apps/rbs02.dbf
cp /u13/oradata/apps/rbs03.dbf /u13/oradata/apps/rbs03.dbf
cp /u14/oradata/apps/system01.dbf
/u14/oradata/apps/system01.dbf
cp /u14/oradata/apps/system02.dbf
/u14/oradata/apps/system02.dbf
#
# *** Create control file
# Modifications:
# 1. Replace database file 'quotes' from '<' and '>' to '''.
# 2. Remove the last reuse log comma.
# 3. Change the <<apps>> in the database file name to <<dba>>
#
ORACLE_SID=dba;export ORACLE_SID
svrmgrl lmode=y <<EOF
connect internal
shutdown immediate
startup nomount
create controlfile set database dba
logfile '/u01/oradata/dba/redo01a.log' size 2m,
'/u01/oradata/dba/redo02a.log' size 2m

```

```

resetlogs
datafile
</u04/oradata/apps/crpd01.dbf> reuse,
</u04/oradata/apps/crpd02.dbf> reuse,
</u04/oradata/apps/ecd01.dbf> reuse,
</u04/oradata/apps/engd01.dbf> reuse,
</u04/oradata/apps/fad01.dbf> reuse,
</u04/oradata/apps/gld01.dbf> reuse,
</u04/oradata/apps/gld02.dbf> reuse,
</u04/oradata/apps/interim01.dbf> reuse,
</u04/oradata/apps/invx01.dbf> reuse,
</u04/oradata/apps/invx02.dbf> reuse,
</u04/oradata/apps/mrpx01.dbf> reuse,
</u04/oradata/apps/mrpx02.dbf> reuse,
</u04/oradata/apps/nohup.out> reuse,
</u04/oradata/apps/pox01.dbf> reuse,
</u04/oradata/apps/rgx01.dbf> reuse,
</u04/oradata/apps/shd01.dbf> reuse,
</u13/oradata/apps/nohup.out> reuse,
</u13/oradata/apps/rbs01.dbf> reuse,
</u13/oradata/apps/rbs02.dbf> reuse,
</u13/oradata/apps/rbs03.dbf> reuse,
</u14/oradata/apps/nohup.out> reuse,
</u14/oradata/apps/system01.dbf> reuse,
</u14/oradata/apps/system02.dbf> reuse,
maxdatafiles 1022
maxlogfiles 32
maxlogmembers 5
maxinstances 1
/
alter database open resetlogs;
EOF

```

## 9. Edit the crdba.sh file to make it work

- change the cp command destination to the previously identified location for each file.
- change the < > brackets in the create control file section to single quotes.
- Change the sid name in the create control file section and in the second half of the copy to reflect the name of the new database.

## 10. Go to the

\$ORACLE\_BASE/admin/new\_sid/pfile directory and edit the initsid.ora and configsid.ora files to reflect the new location of the control files and log files. Also edit them for the new db\_name.

## 11. Go to \$ORACLE\_HOME/dbs and link the newly edited initsid.ora to the current directory. i.e. "ln -s \$ORACLE\_BASE/admin/sid/pfile/initsid.ora initsid.ora".

## 12. Remove the exit at the top of the script.

## 13. Run the script to completion.

## 14. Restart the original instance.

As noted in the header of the script there is no warranty expressed or implied. It is wise to practice this prior to doing anything that may cause a problem.

## Appendix A - Maintaining Database Objects

### Invalid Objects

#### Overview

Oracle uses packages and stored procedures extensively throughout the Apps system. In a more generic sense it refers to them as objects. Other objects are tables, indexes, synonyms and views. Each object is defined in the DBA\_objects view under the system schema. Looking at this view it is possible to determine what schema owns an object, what type of an object it is and whether it is valid. The following SQL code when executed under the system schema will provide the information on any object in the Oracle RDBMS.

```

Chk_objs.sql
column owner format a10
column object_name format a30

select owner, object_name, object_type,
status
from dba_objects
where object_name = '&object_name'
```

#### Output

```

SQL> @chk_objs
Enter value for object_name: gl_interface
```

```

OWNER      OBJECT_NAME
OBJECT_TYPE STATUS
-----
GL         GL_INTERFACE
TABLE     VALID
APPS      GL_INTERFACE
SYNONYM   VALID
```

```

Chk_errors.sql
column name format a15
column text format a50
break on name skip 1 on type
```

```

select name, type, text
from user_errors
order by name, type
/
Output
SQL> @chk_errors
```

```

NAME          TYPE          TEXT
-----
-----
-----
```

```

OEXCPVAL      PACKAGE BODY PLS-00905:
object APPS.MTL_PROJECT_V is invalid
PL/SQL: SQL
Statement ignored
index variable 'LNERR' use is inva
lid
PL/SQL:
Statement ignored
RCV_824_SV    PACKAGE BODY PLS-00201:
identifier 'EC_APPLICATION_ADVICE_PUB.C
REATE_ADVICE'
must be declared
PL/SQL:
Statement ignored
identifier 'EC_APPLICATION_ADVICE_PUB.C
REATE_ADVICE_LINE' must be declared
PL/SQL:
Statement ignored
identifier 'EC_APPLICATION_ADVICE_PUB.C
REATE_ADVICE_LINE' must be declared
PL/SQL:
Statement ignored
12 rows selected.
```

### Compile Invalid Objects

Periodically it is necessary to recompile objects. There are several ways to do this. Aadmin has a function "Compile objects" that helps. It works better in 10.7 than in 10.6. It does not always work completely. I obtained a script from Oracle support that recompiles invalid objects. It compiles objects in order so the objects will compile in as few as runs as possible. I call it comp\_ord.sql because it compiles objects in order.

#### Comp\_ord.sql

```

REM I really don't like hardcoding passwords
in this script (see decode
REM statement below), but all the other
workarounds I've tried failed.
REM There may be some RDBMS bugs in v7.1
which prevent you from performing
REM all the recompilations while connected as
SYSTEM
```

```

connect system/sshields
```

```

column owner noprint
column object_name noprint
column order_col noprint
column owner format a1
column object_name format a1
column order_col format a1
column cmd format a132
```

```

set heading off
set pagesize 0
set linesize 180
set echo off
set feedback off
```

```

spool recomp-sql.sql

select 'set echo on' from dual;

select 'spool recomp-sql.lis' from dual;

select distinct 'connect ' || owner || '/' ||
decode(owner, 'SYS', 'SSHIELDS',
'SYSTEM', 'SSHIELDS',
'APPDEMO', 'APPDEMO',
'APPLSYS', 'APPS',
'APDEMO', 'AP',
'ARDEMO', 'AR',
'PODEMO', 'PO',
'BOMDEMO', 'BOM',
'INVDEMO', 'INV',
'APPDEMOPUB', 'PUB',
'WIPDEMO', 'WIP',
'AKDEMO', 'AK',
'AXDEMO', 'AX',
'CEDEMO', 'CE',
'CNDEMO', 'CN',
'CRPDEMO', 'CRP',
'ECDEMO', 'EC',
'ENGDEMO', 'ENG',
'GLDEMO', 'GL',
'HRDEMO', 'HR',
'JGDEMO', 'JG',
'MRPDEMO', 'MRP',
'OEDEMO', 'OE',
'QADEMO', 'QA',
'RGDEMO', 'RG',
'CSDEMO', 'CS',
'FADEMO', 'FA',
'MFGDEMO', 'MFG',
'OSMDEMO', 'OSM',
'OTADEMO', 'OTA',
'PADEMO', 'PA',
'APPLSYS PUB', 'PUB',
owner) ||

';' cmd,
owner,
1 order_col,
NULL object_name
from dba_objects
where status = 'INVALID' and
object_type in
('TRIGGER', 'PACKAGE', 'PACKAGE
BODY', 'VIEW', 'PROCEDURE', 'FUN
CTION')
and (object_type <> 'PACKAGE BODY' or
(owner, object_name) not in (select
owner, object_name
from dba_objects where
object_type = 'PACKAGE'
and
status = 'INVALID'))
union
select 'ALTER ' ||
decode(object_type, 'PACKAGE BODY',
'PACKAGE', object_type) || ' ' ||
owner || '.' || object_name || ' COMPILE'
||
decode(object_type, 'PACKAGE BODY', '
BODY', '') ||
';' cmd,
owner,
2 order_col,
object_name
from dba_objects
where status = 'INVALID' and
object_type in
('TRIGGER', 'PACKAGE', 'PACKAGE
BODY', 'VIEW', 'PROCEDURE', 'FUN
CTION')
and (object_type <> 'PACKAGE BODY' or
(owner, object_name) not in (select
owner, object_name
from dba_objects where
object_type = 'PACKAGE'

```

```

and
status = 'INVALID'))
order by 2,3,4
/

select 'exit;' from dual;
spool off;

set heading on
set feedback on
set pagesize 9999
set linesize 80

@recomp-sql

```

The comp\_ord.sql works dynamically and compiles all objects that it can. It must be run repeatedly, generally 2-3 times, until it is no longer able to successfully compile additional objects. At this time the dba must take each schema and determine the reason the objects will not compile. As noted in the script the passwords are hard coded so this script may need modified on a system by system basis.

## Stored Packages & Procedures

There are a few “rules” or guidelines that a DBA needs to understand in order to work successfully with packages.

1. Anytime new source code for an object is compiled all objects that call the new object are immediately statused as invalid.

If the new source code has changed any of the calls then any object that calls that part of the new source code must be modified to reflect the change. The objects making the call will remain invalid until they are modified.

2. To compile packages use the comp\_ord.sql program provided by Oracle support. This will try to compile any invalid object. This program will probably have to be executed more than once. This is because proc A may call proc B and proc B is not compiled yet. Once proc B is compiled then proc A will compile.

Understanding these guidelines will help to understand when the packages do not “work” as planned. The following instructions walk through the steps necessary to compile all the packages and procedures. This includes the steps needed after the comp\_ord will not work on it’s own.

1. If you have problems with any of the packages not compiling without error do the following:

- determine what schema is having the problem (usually apps<sup>9</sup>). Use the `chk_objs.sql` script to do this.
- go into SQL\*Plus and connect to the schema
- execute: the `chk_errors` script
- spool the list and print it.

2. The error list will tell you what packages etc. are invalid and why. Generally if a package won't compile without errors it is due to another package. The other package may be valid but the versions of the two packages are incompatible. To find the compatible versions and load them in the APPS perform the following:

- Determine which schema the package should be in. Log into SQL\*Plus as system and run `@ck_objs`. `Ck_objs` will return a list of all the objects in the system with the name given and it's status. Ignore the synonyms if they exist and find the schema/owner that owns the package. This is the schema we want to use.
- `cd $SCHEMA_TOP`. (This is the schema just determined, i.e. AR, BOM.)
- use the UNIX command "find" to determine what UNIX file contains the package definition.

```
find $SCHEMA_TOP -name '*.pls' -exec grep -i
package_name {} \; -print.
```

This command will find all patterns in the .pls files that match the `package_name`. There may be many of them. Look for the one that has "Procedure `package_name`" in it. At the end of the patterns there will be a file name. This is the file that is needed. The `new_file_name` will usually end in `new_file_nameS.pls` or `B.pls`. These files are a set and both are required to put the new package on the system.

- Determine if the file found is the only file on the system with that name. To determine this use the UNIX find command

```
find $SCHEMA_TOP -name 'new_file_name*. -
print.
```

This will return all files in the schema that have the name of `new_file_name`. Including the S and B files.

---

<sup>9</sup> Each schema owns it's own tables but apps owns all the packages and has synonyms to all tables owned by each schema.

- If there are multiple S and B files they must be checked to determine which is the new version. To do this use the UNIX command "more"

`cd` to the directory that contains the S&B set of files.  
`more new_file_nameS.pls`

This will scroll the text of the `new_file_nameS.pls`. Proceed until the version number of the .pls file is shown and record it.

Repeat the procedure for the `new_file_nameB.pls` file.

Proceed to the next directory containing the `new_file_name` and repeat the instructions above until all the versions have been recorded.

- Go to the directory with the highest version number and run the file with SQL\*Plus as follows:

```
SQL*Plus schema/schema @new_file_nameS.pls
SQL*Plus schema/schema @new_file_nameB.pls
```

It is extremely important that S file is run first. If it is not the B file will not compile properly.

- Run `comp_ord` again and see if the original package compiles correctly. If so proceed to the next problem. If the same problem exists for the package try the exercise with the original package. Continue this process until all packages are compiled.

**\*\*Note:** Running the pls files can cause other packages to be labeled as invalid by Oracle. When a new version of a package is installed, Oracle automatically invalidates all packages and views that call the new package. This eliminates the problem with the packages getting out of sync and no one knowing it.