Instance Cloning Under Your Control:
Taking the Mystery Out of Replicating 11i Instances

By James H. Lui

Editor's Note: The best way to avoid "cloning anxiety" is to turn to those who have been down that road before. We asked James Lui to give us his view on demystifying the process for Oracle 11i environments. The result is an informative piece that discusses the challenges of replication across production and development environments, the best tools to use, and the scripts to make it a seamless exercise. James' master and slave scripts are enough for two database articles, so we posted them to our Website www.ORAtips.com Document Library under Database - Cloning.

This is the white paper version of the original article. It is the same as the article, except it has an additional 11 Attachments with scripts.

Executive Summary

While there are many excellent commercial products and utilities available to support E-Business Suite instance replication, including Oracle's own ADClone utility, understanding how these tools function is an integral part of taking command and comprehending the underlying complexity of the architecture of the E-Business Suite. Duplication of a production E-Business Suite environment is a common need during creation and maintenance of both development and standby (disaster recovery) environments.

Using common SQL scripts and common UNIX (and Windows-compatible) tools, utilities, and techniques, this presentation will take you "under the covers" of the world of the Java and GUI tools, and into control and configuration of the E-Business Suite technology stack using command-line level resources. In short, we take a look at what cloning an instance really means, and how to extend the process and apply your own particular business requirements.

Within this article, we present a solution based upon a live production to development and standby replicated environment. You will learn how to identify and control critical components of the Oracle E-Business Suite Technology Stack. You will understand how to adapt the presented scripts and utilities to your own environment. Finally, you will learn what the limitations are to this approach and when you should seek 3rd-party tools.

I. The Challenge

Refreshes

We all know it's bad to test out newly developed code or bug fixes using a production instance – that's why we have development instances around. But a frequent complaint of most development groups is that the instances used for such testing are often filled with stale data that bears little resemblance to the production environment. So while something works perfectly well in development, it bombs when moved to production, usually because the table data is any number of days more current than what was used for testing.

Even on its own merit, the Oracle E-Business Suite (EBS) has continuing releases of patches that need to be tested against production data prior to deployment. Thus, a method of rapidly creating a living duplicate of the production environment becomes immediately apparent for any customer using EBS. Continuing to proceed manually will eventually impact testing schedules, and you'll end up searching for other ways to save time or resources to continue to deliver a consistent product.

$FND_TOP/bin/adclone (the administrative utility provided with EBS) has evolved over time to provide a convenient end-to-end full replication of the entire stack contents from database to presentation layer components. Sometimes that type of replication is much more than what is actually needed for testing. Additionally, the prerequisites for using *adclone* include having significant extra file system space available to replicate, and then restore the information and files on all of the tiers, so sometimes *adclone* is just not a practical solution for many customers.

For example, if you know your test instance $APPL_TOP code tree is already reasonably synchronized with your production environment and you haven't applied any other patches to it, or modified it other than for the purposes of your testing, why would you want to execute a full replication of the whole stack if not needed? You just want a copy of the production data available for your use. This would be roughly equivalent to wanting to test a new brand of gasoline, but needing to re-create the entire car and engine to do it – instead of just draining the gas tank and refilling with the test fuel.

Or perhaps the kind of testing you're performing is "destructive", such as General Ledger move-merge operations, so it can only be performed once against a given set of data. All you're seeking is a simple copy of the production data that's reasonably up-to-date, but configured for testing purposes. Every DBA ends up learning how to replicate a copy of a database at some point. Included in that process is usually a basic renaming of the instance (changing the SID), and sometimes resetting the database SYSTEM and SYS account passwords. But entering the highly inter-dependent world of the E-Business Suite means you can't just ALTER USER the rest of the application passwords without breaking technology stack connectivity.

And that's what this solution was designed to provide – a controlled and relatively automatic method of simply replicating production to a development instance, with security and identity features built-in so that no mistake is made in knowing whether you're logging into production or test.

Consistency

So our goal is some reasonable level of consistency between a development instance and the production systems it is supposed to reflect. To do this, we have a number of different portions of the application system to address:

- Table data
- Program code
- Presentation Layer objects (forms, reports, etc.)
- System components
- OS level components

Each component that is different between environments constitutes another variable towards inconsistent test versus production results. Within this solution, we'll attempt to accommodate replication of the database tier, partial replication of the applications tier, and standardized identity management to provide a reasonable amount of differentiation between the production environment and the resulting development copy.

Well beyond the scope of this solution would be attempting to support consistency across different OS platforms or revisions, requiring partitioning or advanced replication architecture across multiple hosts, or similar situations requiring dependent decisions being made based upon dynamic environment conditions.

Automation

While performing all of the manual steps required when synchronizing all or parts of the application system is possible to do manually, its tedious and repetitive. Manual processes are prone to human errors, and there remains a significant learning curve required to transfer manual process knowledge to cross-trained members of your department. Wherever possible, we will attempt to model the logical decision-making process applicable to how the refresh would occur if being handled manually and provide appropriate error detection and handling to streamline the overall interactive presentation layer of this solution.

II. The Tools

SQL

SQL*Plus is used to extract current environment configuration and reset values that must be changed once the target environment has been re-established. Since the techniques used in this solution are designed to be reusable regardless of EBS version, there is no built-in dependency for any particular version of SQL*Plus as long as it's compatible with the database being configured.

Korn Shell

This technology is how we control processes that either need multiple components to complete a task, or to remotely control functions on other hosts. Programmatically, we seek something that provides iterative control over tasks being automated and some method of providing rudimentary if-then-else decision processing.

Alternatives:

- Perl
- Automation tools (ala Opalis)

Oracle's Command-line Toolkit

Providing built-in functionality to manipulate many attributes of the E-Business Suite. ad<component>ctl.sh and ad<component> executables:

- *adstrtal.sh* – Controls startup of registered services for the EBS stack.
- *adstpall.sh* – Shuts down said registered services for the EBS stack.
- *$AD_TOP/bin/FNDCPASS* – Allows changing of passwords at the database and application level

What about Windows?

The objective of adopting this methodology is to simulate the UNIX OS capabilities under Windows. Some options:

MKS Toolkit (commercial) or Cygwin (GNU public)

- Creates a UNIX shell environment that provides similar Korn and Bourne scripting and file system emulation for the Windows environment.

Good Old Batch language (BAT or CMD scripts)
You can perform much of the functionality found using the Korn shell scripting in standard batch language. But this would involve more hard-coding and later corrections than would be desirable, and you'll lose much of the dynamic nature of these scripts unless you use some sort of

algorithm-supporting extension or supplement, such as Perl, and involve more hard-coding and later corrections than would be desirable.

III. The Solution Architecture

The Master Script

refresh_master.ksh – The one-stop "does everything but the kitchen sink" control script.

Originally, each step of the process was developed incrementally. Simple SQL scripts were created to reset passwords and control badly behaving alerts; and small Korn shell scripts were designed to handle the startup, shutdown, and recovery of the instance. But as the list of scripts to execute continued to grow, the need to automate and link the procedures became readily identifiable.

The master script is designed with modularity of purpose and extensibility in mind because it is not completely dedicated to singular use with the E-Business Suite. This script can and has been used for refreshes of any of our Oracle-databases that are based upon the same architectural standards for file system-level configuration as deployed for the EBS instance. Whether using external flag files, or through utility-based parsing of an external file containing instance identity information (such as an *init.ora* or placing comments within a "recreate control file" SQL script), this script can be modified to determine the type of database it is replicating, and thus establish logical breakpoints in the code to enable or ignore various sub-functions within it.

The "Slaves"

As mentioned previously, this solution was developed incrementally, so each sub-routine script was designed to be executable independently of each other.

refresh_notice.ksh – provides a means of notifying users and other developers that a refresh is pending and that any cancellation requests should be made immediately. Also serves to provide an additional audit trail item for refresh activities.

conc_mgr_control – in 10.7 and 11i versions through 11.5.8, some of the middle-tier processes are not subject to centralized control, such as via the *adstrtal.sh* and *adstpall.sh* scripts. This script was developed originally in 10.7 to provide an automated method of starting and stopping the Concurrent Managers. As evolved in 11.5.8, it now controls the startup and shutdown of the adstrtal.sh/adstpall.sh scripts and the Workflow Mailer processes.

db_control.ksh – a relatively simple script to start or shutdown an instance. This is where you might include any extra steps desired, such as log file switching or control file backups before the instance is actually stopped. Also you could include supporting standby database startups, SGA package pinning, and other preparatory steps necessary as part of instance startup.

fnd_setup – we support up to 10 development instances on the development host systems. A simple but effective Korn shell menu selector is used to switch among the various available environment setups to present the various choices to the user.

Remote Controls

rsh – currently used to kick-off the various scripts found on the middle-tier host. This is not required for single-tier configurations. SSL technology (SSH) is planned as a replacement for this utility to avoid .rhost file exposure for security purposes.

rcp – used for remote copying between hosts. Could be substituted with FTP or NFS mounts. Also planned for SSL-based substitution in the future (scp).

The "Co-Dependent" Config Files

<SID>.xml – the "master" configuration repository for AutoConfig (*adautocfg.sh*) –this is the critical file used to re-configure the newly restored copy of production data to the specific application settings for the designated development instance.

$APPL_TOP/admin/<SID>.env – environment settings needed to execute the *adstrtal.sh* and *adstpall.sh* scripts (or in 10.7, startup and shutdown of the concurrent manager).

$ORACLE_HOME/<SID>.env – need to be able to instantiate a proper environment for database control via SQL*Plus.

The "No-See-Um's"

Not covered within the scope of this document are the other pre-existing support solutions that provide the fundamental environment supporting this sort of automated refresh.

A pre-staged copy of production database – our daily database backup routine automatically provides 6 days of hot and 1 day of cold database backup from the production instance, pre-staged and compressed on the development host. This is designed to minimize the time normally required to copy the data files from the production to development host and leverage Oracle's point-in-time recovery capability. You can emulate this simply by providing a copy of the data files from your production instance's cold or hot backup in an accessible area to your development host (with or without compression).

A pre-staged copy of production $APPL_TOP code tree – while not in extensive use with 11i since the types of testing we perform no longer require constantly available refreshes of the $APPL_TOP, $OA_HTML, and $ORACLE_HOME branches, we do pre-stage a copy of the production configuration on each box for use both as a potential standby production middle-tier, and for use in copying a fresh branch set based upon the present production configuration. During our 10.7 use, we generated a weekly compressed copy of the complete $APPL_TOP and staged it on the development hosts so that all of our various custom reports, SQL scripts, and forms would be brought over with ease. Portions of the master refresh script continue to support this functionality, if available.

Archive logs – our present configuration uses Oracle's DataGuard (DG) technology to provide replication and recovery of the production database to the standby and development hosts. We take advantage of DG's automatic synchronization of the archive logs (from production) on the development host during the recovery phase of a hot backup-based refresh. Prior to DataGuard, we included automatic archive and transfer of the production archive logs to the development host staging mount point as part of the nightly backup scripts.

Passwords (ora_pass) – we also use a password cloaking system to pre-seed the environment variables used within the scripts (e.g., $APPS_PW) so that hard coding is not necessary. This could also be eventually replaced with a similar Lightweight Directory Access Protocol (LDAP) or Network Information Service (NIS) style authentication system, such as Oracle Internet Directory (OID), if present.

IV. How It Works

The Play-by-Play

refresh_master.ksh –All scripts listed can be found at www.ORAtips.com / Document Library under Database – Cloning.

refresh_master.ksh

- *conc_mgr_control stop {SID}* – Stops the middle-tier components.
- *db_control.ksh stop {SID}* – Shutdown the database.
  - *recr{SID}_hot.sql* (Hot Backups) – Recreate the controlfile and rename the DB.
    global_updates.sql – Perform any desired uniform updates.
    recrOEM.sql – Recreate the orapwd file and restore remote_login_passwordfile.
    passupdt_imp.sql (10.7 obsolete) – Older method to reset development passwords using import of FND tables.
    update_fnduser.sql – Modify specific Apps-related security.
    passchg.ksh (mid-tier) – Execute database and application password changes.
    conc_mgr_control start {SID} – Start the middle-tier components.
  - *recr{SID}.sql* (Cold Backups) – Control file recreation without recovery required.
    global_updates.sql
    recrOEM.sql
    passupdt_noimp.sql (10.7 obsolete)
    update_fnduser.sql
    passchg.ksh (mid-tier)
    conc_mgr_control start {SID}

Log in as your UNIX user ID, then SU as *oracle.*

```
su – oracle
<oracle password>
```

Send the e-mail notice of the refresh. (Specify first and last name of the requestor and how many minutes notice you are providing).

```
cd /oracle/bin
. <SID>_setup (e.g., . PRE_setup)
./refresh_notice.ksh [First Name] [Last Name] [No of Mins to refresh] (Attachment 1)
e.g., ./refresh_notice.ksh John Doe 30
```

Login as *oracle* then execute master refresh script.

```
su - oracle
<oracle password>
cd /oracle/bin
./refresh_master.ksh (Attachment 2)
```

The process begins with a simple e-mail notification to both administrative and development users, as well as any user that has logged into the instance to be refreshed since it was created. This step is kept separate to allow for cancellation based upon customer request (i.e., the instance is still being actively used.)

Once clearance has been received (often by default through no responses to the notification e-mail), the master script is executed:

This script first prompts the user to select an instance from a list of available instances. Then an appropriate backup source is selected. The default selection is the pre-staged weekly cold backup copy already present on the development host staging area. Any alternate backups available on the source production host are also listed (selection of which will trigger a remote copy process to bring that backup into the staging area on development).

The user is prompted to provide a basic description and requestor name to be used during notification after completion. After final confirmation, the refresh process begins. The selected development instance is shutdown and all data files are removed. The staged backup is decompressed (with parallelism of processes available depending on the number of CPUs) and a "recreate control file" script is executed to register any changes in data file location(s) and store the new instance's identity settings. In preparation for use as a development instance, changes in security are made, account passwords expired, various alert triggers dropped or de-activated, and the concurrent processing manager settings are modified.

Finally, passwords are changed throughout the instance at both the database and application levels with appropriate remote login password file modifications executed.

After processing (actual speed will depend on your database size, hardware platform, number, and speed of CPU's, etc.) beginning with the original refresh notice, a completed refresh of the development instance chosen is complete, with all passwords reconfigured for development environment standards, unauthorized user access end-dated and disabled, and initial instance defaults reset for use with the new development configuration specified.

Expected Results

The following is from an actual refresh executed on our Sun Solaris-based development host. User responses are indicated with brackets "[ ]":

MOVM:oracle > [*refresh_notice.ksh James Lui 1*]

```
**************************************
**    PLEASE CHOOSE CAREFULLY!!    **
**************************************
Which Oracle Financials Environment?

APPL_TOP Homes for Refreshes
DEFAULT->
1)  Training              (D107)
2)  Pre-Production        (PRE)
3)  Pre-Production 2      (PRE2)
4)  Staging               (MDD)
5)  Staging2              (STG2)
6)  Test                  (TEST)
7)  Test 2                (TST2)
```

8) FSG Development (FSG)
9) Move/Merge (MOVM)
10) Oracle Homes for Refreshes
11) Training (D107D)
12) Pre-Production (PRED)
13) Pre-Production 2 (PRE2D)
14) Staging (MDDD)
15) Staging2 (STG2D)
16) Test (TESTD)
17) Test 2 (TST2D)
18) FSG Development (FSGD)
19) Move/Merge (MOVMD)

1: [*19*]

E-mail notice sent.

MOVM:oracle > [*refresh_master.ksh*]

****
** PLEASE CHOOSE CAREFULLY!! **
****

Which Oracle Financials Environment?

APPL_TOP Homes for Refreshes
DEFAULT->
1) Training (D107)
2) Pre-Production (PRE)
3) Pre-Production 2 (PRE2)
4) Staging (MDD)
5) Staging2 (STG2)
6) Test (TEST)
7) Test 2 (TST2)
8) FSG Development (FSG)
9) Move/Merge (MOVM)
10) Oracle Homes for Refreshes
11) Training (D107D)
12) Pre-Production (PRED)
13) Pre-Production 2 (PRE2D)
14) Staging (MDDD)
15) Staging2 (STG2D)
16) Test (TESTD)
17) Test 2 (TST2D)
18) FSG Development (FSGD)
19) Move/Merge (MOVMD)

1: [*19*]

-- Page Break --
Select Source Backup to Refresh Database: MOVM

1) Hot Staging

2) Cold b04 Dec 23 16:43
3) Hot b07 Dec 23 02:28
4) Hot b01 Dec 22 02:28
5) Hot b08 Dec 21 02:29
6) Cold b03 Dec 19 16:01
7) Hot b02 Dec 18 02:29
8) Hot b06 Dec 17 02:29
9) Hot b05 Dec 16 02:28

Default (1): [*1*]

-- *Page Break* --

CAREFULLY verify the refresh parameters you have chosen.

Database: MOVM
Passwords: Manual
Backup: Local Staging Directory
Type: [Hot]
Code Refresh: No

Do you wish to proceed with the refresh? (y/n/q): [*y*]

-- *Page Break* --

Enter First Name of person requesting refresh: [*James*]
Enter Last Name of person requesting refresh: [*Lui*]
Describe the purpose of this refresh: [*11.5.10 Upgrade 2 Refresh*]
List the timestamp of the refresh source: [*24-FEB-2005 15:00*]
This instance is being used by James Lui for the purpose of: 11.5.10 Upgrade 2 Refresh
This will result in a refresh of data as of: 24-FEB-2005 15:00

Okay to proceed with the refresh? (y/n/q): [*y*]

-- *Page Break* --

17:49: Initiating refresh of MOVM using hot backup.
17:49: Stopping ConcMgrs.
17:49: Stopping Database.
17:49: Clearing application and database logs/reports.
17:49: Building sample serial uncompress script.
17:50: Removing current data files.
17:50: Running uncompress script with 4 worker(s). (be patient)
17:50: Datafile restoration portion began.
18:21: Datafile restore portion ended.
18:21: Calling db recreate script: /oracle/admin/bin/refresh_slave01.ksh (Output captured in /oracle/admin/bin/MOVM_slave01.log)
18:34: Performing automatic password reset...
18:36: SYSTEM and SYS user passwords changed to defaults for MOVM
18:36: Changing passwords to defaults for MOVM - REAL mode.
18:36: Application User password update script created - /oracle/admin/passupdt/passupdt.ksh
18:36: Executing... /oracle/admin/passupdt/passupdt.ksh

Log filename : L1695116.log

Report filename : O1695116.out
<...snip... repeats depending on number of schemas being reset>

18:39: Log file /oracle/admin/passupdt/passupdt.log generated.
18:39: Checking for Errors....
18:39: MOVM updated back to default passwords.
18:39: Executing AD Auto Configure (respond with APPS pw when prompted...

APPL_TOP is /oracle/movmappl
ORACLE_SID is
TWO_TASK is MOVM
ORACLE_HOME is /oracle/movmora/8.0.6

Enter the APPS user password:
[<*APPS password*>]

Using AutoConfig to configure the Applications environment

Loading APPL_TOP environment from /oracle/movmappl
Using context: /oracle/movmappl/admin/MOVM.xml

Context Value Management will now update the context file
Updating context file ... COMPLETED

Configuring templates from all of the product tops ...
Configuring AD_TOP.....COMPLETED
Configuring FND_TOP.....COMPLETED
<...snip... repeats depending on number of schemas being reconfigured>
Configuring XNC_TOP.....COMPLETED
Configuring WSH_TOP.....COMPLETED

The log file for this session is located at:
/oracle/movmappl/admin/MOVM/log/02250639/adconfig.log
18:39: Auto Configure of MOVM complete.

18:39: ConcMgrs can be started now (theoretically).
18:39: Starting ConcMgrs.
18:39: Refresh of MOVM complete.
18:39: Sending e-mail refresh complete notice.

MOVM:oracle >

Anticipated Things To-Do Afterwards

FNDFS_ profile option (RRA: Service Prefix) reset – when multiple instances of EBS share the same host, a system profile option needs to be set in order to distinguish incoming FND File Server requests among the various instances.

Custom code updates – modify any customized function definitions for new hostnames. In order to provide easy and direct accessibility by users, some of our specific Discoverer pages have been registered as functions. These usually are transferred with specific hostname: port assignments intact and you need to adjust accordingly for the new host.

$ORACLE_HOME updates – when specific database kernel patches are being tested in development without a segregated $ORACLE_HOME, you will need to re-apply any code or library updates to maintain compatibility with the patched kernel (e.g., Security Alert 68 replaced much of the Oracle Web Applications packages that needed to be re-loaded via a re-execution of *owaload.sql*).

Oracle Portal integration – with a number of different versions of Portal in simultaneous support by any given version of EBS, some will self-configure using adautocfg.sh and others will require manual adjustment after the refresh to restore functionality.

Recompile invalid objects - once all of the final adjustments have been made, you should recompile any invalid objects to ensure a consistent baseline data set for your instance compared with production.

Exploring the Possibilities

Integrating Other Ideas

Update FND_PROFILE_OPTION_VALUES automatically – through a combination of passing through appropriate values for hostname, SID, port numbers, and other values that can be extracted from the <SID>.xml repository, a relatively simple parameterized SQL script could be used to update any non-AutoConfig controlled settings during the post-refresh stages.

Data scrambling – at one point, we were using Oracle HRMS as our primary personnel information source. This mandated development of several scripts designed to scramble and mask various pieces of sensitive employee information (e.g., social security numbers/national identifiers, birth dates, private contact information). These scripts were generally incorporated as part of the *global_updates.sql* script for consistent use during all refresh processes.

Cloaking and Secrecy

Hard-coded script passwords are a pretty big security loophole. In order to deter intruders, but provide continued non-interactive and flexible operation, we could consider the possibility of securing our transaction scripts in a few different ways:

- 3rd party authentication
- Multi-factor authentication – supported at the OS, database, and application levels.
- Oracle Internet Directory/LDAP/NIS
- Lightweight secrecy using the OS

This final option (OS-level cloaking) is in present use while the rest of our enterprise-level LDAP implementation continues deployment.

Cold Fusion...

Automatically verify proper process launching on middle-tier (i.e., ps –ef | grep ${SIDLC}| wc –l) – presently the final portions of the solution blind launch the middle-tier components, which could fail if the database refresh did not complete successfully. A check for the appropriate number of components running (including the expected timestamps so as to address possible zombie

processes) would be useful to prevent final success notification upon detection of this kind of error.

Control de-activation and re-activation of development instance availability on Portal – the present configurations of Portal 3.0.9 and 10g will only return a "System Maintenance" error page, which is the same whether a refresh is in process, or a complete failure of the instance has occurred. A preferred method would be to deactivate the development portlet while unavailable during refreshes.

Increase complexity of notification and workflow features to enable complete hands-off operation – currently the system sends a simple e-mail notifying completion of the refresh process. A preferred enhancement would be notifications to be sent at any interrupted stage of the process, including some sort of diagnostic information and any recommended correction procedures.

Automatically generate instance selection menus based upon /var/oracle/oratab contents – currently this is manually coded and maintained.

The PARALLEL decompression sub-function – presently performs a simple directory listing of available files to derive its master file list. This list is passed to (n) number of separate environment variables to divide the list into a relatively equal number of files for each decompression daemon to process. Load balancing of the decompression routine could be accomplished by pre-sorting the file list by size. Then the list could be split into the appropriate number of sub-lists. You could also incorporate a self-configuring feature to limit the number of separate sub-lists to the number of CPU's found on the executing host.

Conclusion

With this type of solution, designed for extensibility, flexibility, and dynamic response to changes in the physical or logical environment, the process of maintaining development environments (or even replicating production to several standby systems) becomes a simplified process, easy to train others to use without compromising security issues between production and development areas, and even provides a complete audit trail for purposes of determining what is actually being performed during the entire refresh process. By studying the individual stages of change within the master and each of the slave scripts, this solution becomes a useful training tool for educating others how each of the steps works and what each step does functionally.

About the Author

James Lui, Employers Insurance Group. - James has been a primary implementer and end-user of the Oracle E-Business Suite (EBS) for over 12 years from virtually every aspect of use and control including process documentation, gap analysis, forms and reports development, and the many diverse formal roles including database administrator (DBA) and Apps System Administrator. His expertise parallels the evolution of EBS from the 10.7 character client through the latest 11i releases, including "Controlled Release" modules such as Enterprise, Planning and Budgeting (EPB). He has remained completely hands-on with every implementation, upgrade, and migration project with particular focus on process simplification, management automation, and getting EBS to perform to customer expectations and business needs based upon limited resources. James may be contacted at James.Lui@ERPtips.com.

Attachment 1 *refresh_notice.ksh*

```
#! /bin/ksh
#
# 20-JAN-99, Lee Turner
# 9-MAR-99, J. Lui - added Instance selection menu
# 10-NOV-02, T. Marlow - added 8i v. 7.x version checking
#
# This script generates an email message to notify users of a pending
# database refresh.  You must have your environment pointing to the
# database to be refreshed and the database must be running.  Script is
# not account dependent.
#
# Command Line Parameters:
# First_Name, Last_Name, Minutes, <Hours/Days/other unit>
#
# Check the user id.
if [ -x /usr/bin/id ]
then
        CHKID=`id | sed 's/[^a-z0-9=].*//'`
        if [[ $CHKID != "uid=101" ]]; then
                echo "$0:  Only oracle can run $0"
                exit 1
        fi
else
        echo "$0:  Unable to find /usr/bin/id"
        exit 1
fi

if [ $# -lt 3 ]; then
  echo "\nERROR: Usage: $0 [first_name] [last_name] [minutes]\n"
  exit
fi

FNAME=$1
LNAME=$2
QTY=$3

if [ $# -eq 4 ]; then
  TMEAS=$4
else
  TMEAS="minutes"
fi

export ORAENV_ASK=NO
. /usr/lbin/fnd_setup
. /usr/lbin/ora_pass

if [[ $ORACLE_SID = "" ]]; then
  OSID=$TWO_TASK
else
  OSID=$ORACLE_SID
```

```
fi
if [ `grep $OSID /etc/oratab | wc -l` -eq 0 ]; then
        echo "\n  ERROR: Invalid or unset ORACLE_SID.\n"
        . /usr/lbin/fnd_setup
fi


SUBJECT="Refresh of Sun $OSID database pending!"
MESSAGE=/usr/tmp/.refresh_message
DATABASE="v\$database"
DATE_RANGE="WHERE to_date(substr(b.created,1,9),'DD-MON-RR') <=
to_date(a.last_logon_date,'DD-MON-RR')"
# Uncomment and replace for version 7.x databases
# DATE_RANGE="WHERE to_date(substr(b.created,1,8),'MM/DD/RR') <= a.last_logon_date"

#
# Set FinSys email list.
#
FINSYS_EMAIL="me@me.com"

#
# Get list of email address for users outside of FinSys who have logged in
# since the last refresh.
#
USERS_EMAIL=`sqlplus -s apps/$APPS_PW << EOF
SET    HEAD  off
SET    FEED  off
SET    ECHO  off
SET    PAGES 0
COL    email for a40 trunc
SELECT a.email_address email
FROM   fnd_user a, $DATABASE b
$DATE_RANGE
AND    a.user_name NOT IN ('SYSADMIN','JLUI')
AND    a.email_address IS NOT NULL
AND    a.end_date IS NULL;
exit
EOF`

#
# Get list of application accounts that have logged in since the last refresh.
#
USERS=`sqlplus -s apps/$APPS_PW << EOF2
SET    HEAD  off
SET    FEED  off
SET    ECHO  off
SET    PAGES 0
COL      ldate for a10 trunc
COL      uname for a20 trunc
SELECT   to_char(a.last_logon_date,'DD-MON-RR') ldate, a.user_name uname
FROM     fnd_user a, $DATABASE b
$DATE_RANGE
AND      a.user_name NOT IN ('SYSADMIN')
```

```
AND      a.end_date IS NULL
ORDER BY 2;
exit
EOF2`

#
# Get last date of refresh.
#
RDATE=`sqlplus -s apps/$APPS_PW << EOF3
SET    HEAD  off
SET    FEED  off
SET    ECHO  off
SET    PAGES 0
SELECT   created
FROM     $DATABASE;
exit
EOF3`

#
# Generate message.
#
echo "\nA refresh of the $OSID database has been requested by $FNAME $LNAME and will" >
$MESSAGE
echo "begin in $QTY ${TMEAS}." >> $MESSAGE
echo "\n$OSID was last refreshed:  $RDATE" >> $MESSAGE
echo "\nThe following users have logged into the database since the last refresh:\n\n\tLast
Login\tUser\n\t----------\t--------" >> $MESSAGE

CNT=1
for i in $USERS
do
  if [ $CNT -eq 1 ]; then
    echo "\t$i\c" >> $MESSAGE
    ((CNT=$CNT+1))
  elif [ $CNT -eq 2 ]; then
    echo "\t$i" >> $MESSAGE
    CNT=1
  fi
done

echo "\nCONTACT FINANCIAL SYSTEMS IMMEDIATELY IF THIS PRESENTS A PROBLEM
FOR YOU!\n" >> $MESSAGE

#
# Send message.
#
cat $MESSAGE | mailx -s "$SUBJECT" $FINSYS_EMAIL $USERS_EMAIL
# Uncomment following for debugging
# echo "APPS_PW is $APPS_PW"
# echo "OSID is    $OSID"
# echo "mailx -s "$SUBJECT" $FINSYS_EMAIL $USERS_EMAIL\n"
# cat $MESSAGE
#rm $MESSAGE
```

Attachment 2 *refresh_master.ksh*

```
#!/bin/ksh
#
# Name         :  refresh_master.ksh
# Author       :  James Lui
# Updated      :  05-18-2000  J. Lui - Added Refresh E-mail code at end.
#                 09-28-2000  J. Lui - 8i Compatibility
#                 06-20-2003  J. Lui - Parallel uncompression
# Documentation:  SiteDocs.doc on P:\Oracle\Site Docs and Documentation
#
#*******************************************************************************#

# Check the user id.
#if [ -x /usr/bin/id ]
#then
#       eval `id | sed 's/[^a-z0-9=].*//'`
#       if [ "${uid:=101}" -ne 0 ]
#       then
#               echo "$0:  Only oracle can run $0"
#               exit 1
#       fi
#else
#       echo "$0:  Unable to find /usr/bin/id"
#       exit 1
#fi
    #*********************************************************************#
    #                                                                     #
    #                  V A R I A B L E S                                  #
    #                                                                     #
    #*********************************************************************#

PARALLEL=4
BBIN=/usr/lbin/backup/cold/bin
STAGE=/stage/oracle/backups/PROD/stage
REDO=/stage/oracle/backups/PROD/redo
CODE_TAR="/u01/applmgr/common/prod_code/prod_appl_top.tar.Z"
OBASE=/oracle/admin

export ORAENV_ASK=NO

    #*********************************************************************#
    #                                                                     #
    #                  F U N C T I O N S                                  #
    #                                                                     #
    #*********************************************************************#


################
#
function error_test {
  CALLING_FUNC=$1
  ERR_CALL=$2
```

```
  if [ $ERR -eq 1 ]; then
    if [ $ERR_CALL -eq 0 ]; then
      echo "\n           '${ENV}' is an invalid selection."
      echo   "           Please choose again.\n"
      sleep 3
    fi
    $CALLING_FUNC
  fi
}
# End Function 'error_test'
################


################
#
function set_backup_source {
#
#  This function determines and sets variables for the backup source to
#  be used and for the type of backup (hot or cold).
#
#    Key Variables:
#    --------------
#    COPY_DIR        "NULL" = Using Local Stage Directory
#                    bXX = Remote Directory Name
#
#    STAGE_FLAG           0 = Using Remote Directory
#                         1 = Using Local Stage Directory
#
#    BU_TYPE_FLAG         0 = Cold
#                         1 = Hot
#

#
#  Initialize variables...
#
ITR=2
CNT=1

#
#  Determine whether the backup in the staging directory is hot or cold...
#
if [ -f ${REDO}/* ]; then
  STG_BU_TYPE="Hot"
else
  STG_BU_TYPE="Cold"
fi

tput clear
echo "\nSelect Source Backup to Refresh Database:  $OSID"
echo "\n  1)\t${STG_BU_TYPE}\tStaging"

#
#  List backup dirs, types (hot or cold), and time stamps in Production
```

```
#
for i in `rsh cagln1s111 ls -ltd /nearline/backup/PROD/b*`
do
  if [ $CNT = 6 ]; then       # Set Month
    MON=$i
    ((CNT=$CNT+1))
  elif [ $CNT = 7 ]; then     # Set Day
    DAY=$i
    ((CNT=$CNT+1))
    if [ `echo $DAY | awk '{print length($0)}'` -eq 1 ]; then
      DAY="0$DAY"
    fi
  elif [ $CNT = 8 ]; then     # Set Time
    TIME=$i
    ((CNT=$CNT+1))
  elif [ $CNT = 9 ]; then     # Set Directory and Backup Type
    RMT_BU_TYPE=`rsh cagln1s111 "if [ -d ${i}/redo ]; then
      echo Hot
    else
      echo Cold
    fi
    "`
    eval "RMT_BU_TYPE${ITR}=${RMT_BU_TYPE}"

    BDIR=`basename $i`
    eval "P_ITR${ITR}=${BDIR}"
    echo "  ${ITR})\t${RMT_BU_TYPE}\t${BDIR}\t${MON} ${DAY} ${TIME}"
    ((ITR=$ITR+1))
    CNT=1
  else
    ((CNT=$CNT+1))
  fi
done

echo "\nDefault (1): \c"; read CHOICE

if [ `echo $CHOICE | wc -m` -eq 1 ] || [ $CHOICE -eq 1 ]; then
  COPY_DIR="NULL"
  STAGE_FLAG=1
  BU_TYPE=$STG_BU_TYPE
else
  eval "COPY_DIR=\$P_ITR${CHOICE}"
  STAGE_FLAG=0
  eval "BU_TYPE=\$RMT_BU_TYPE${CHOICE}"
fi

#
# If User attempts to select a number other than those listed, error
# and repeat selections.
#
if [ `echo $COPY_DIR | awk '{print length($0)}'` -eq 0 ]; then
  echo "\n\nERROR! Invalid choice! Select again.\n"
  sleep 3
```

```
  set_backup_source
fi

if [ "$BU_TYPE" = "Hot" ]; then
  BU_TYPE_FLAG=1
else
  BU_TYPE_FLAG=0
fi

}
# End Function 'set_backup_source'
################


################
#
# Force manual until revised.
function set_pw_reset_flag {
  ERR=0
  ERR_CODE=0
  ENV=1
#  tput clear
#  echo "\nPlease select the method to use for password updates:\n"
#  echo "      1)  Automatic"
#  echo "      2)  Manual\n"
#  echo "Default (${ENV}):  \c"
#  read ENV
#  if [ `echo $ENV | wc -m` -eq 1 ]; then
#    ENV=1
#  fi
#  echo ""
#  case $ENV in
#    1)  PW_RESET_FLAG=0;;
#    2)  PW_RESET_FLAG=1;;
#    *)  ERR=1;;
#  esac
  PW_RESET_FLAG=1
#  error_test set_pw_reset_flag $ERR_CODE
}
# End Function 'set_pw_reset_flag'
################

################
# Do not control APPL_TOP refresh at this time.
function set_code_flag {
  ERR=0
  ERR_CODE=0
  ENV=N
#  tput clear
#  echo "\n  Do you want to refresh the application code (APPL_TOP) as part"
#  echo   "  of the database refresh?"
#  echo "\n  WARNING!!!"
#  echo   "      1)  Make sure all customized code is preserved."
```

```
#  echo   "      2)  This adds approximately 1 hour to the refresh process."
#  echo "\n  y/n (n): \c"
#  read ENV
#    if [ `echo $ENV | wc -m` -eq 1 ]; then
#    ENV=N
#  fi
#  case $ENV in
#     y|Y)
#          if [ -f $CODE_TAR ]; then
#            CODE_FLAG=1
#
#           echo "\n  ERROR: Cannot locate application code image:"
#            echo   "         $CODE_TAR\n"
#            echo   "         You must either skip the code refresh or"
#            echo   "         put the image file in place.\n"
#            sleep 5
#            ERR=1
#            ERR_CODE=1
#          fi
#     ;;
#     n|N)  CODE_FLAG=0;;
#     *)  ERR=1;;
#  esac
#
#  error_test set_code_flag $ERR_CODE
  CODE_FLAG=0

}
# End Function 'set_code_flag'
################

################
#
function confirm_refresh {

tput clear
echo "\nCAREFULLY verify the refresh parameters you have chosen.\n"
echo "        Database:  $OSID"

echo "       Passwords:  \c"
  if [ $PW_RESET_FLAG -eq 0 ]; then
    echo "Automatic"
  elif [ $PW_RESET_FLAG -eq 1 ]; then
    echo "Manual"
  fi

echo "          Backup:  \c"
  if [ $STAGE_FLAG -eq 0 ]; then
    echo "Remote from \c"
    rsh cagln1s111 ls -ltd /nearline/backup/PROD/$COPY_DIR | awk '{print $9}'
    echo "           Dated:  \c"
    rsh cagln1s111 ls -ltd /nearline/backup/PROD/$COPY_DIR | awk '{print $6,$7,$8}'
  elif [ $STAGE_FLAG -eq 1 ]; then
```

```
   echo "Local Staging Directory"
  fi

echo "           Type:  [\c"
  if [ $BU_TYPE_FLAG -eq 0 ]; then
    echo "Cold]"
    BTYPE_TEXT="cold"
  elif [ $BU_TYPE_FLAG -eq 1 ]; then
    echo "Hot]"
    BTYPE_TEXT="hot"
  fi

echo "    Code Refresh:  \c"
  if [ $CODE_FLAG -eq 0 ]; then
    echo "No"
  elif [ $CODE_FLAG -eq 1 ]; then
    echo "Yes\t($APPL_TOP)"
  fi

echo "\nDo you wish to proceed with the refresh? (y/n/q): \c"; read ANS
case $ANS in
    y|Y)  tput clear
          RUN_STAT=1;;
    q|Q)  exit;;
     *)  echo "Cycle through setup screens again" > /dev/null
          RUN_STAT=0;;
esac

read FNAME?"Enter First Name of person requesting refresh: "
echo
read LNAME?"Enter Last Name of person requesting refresh: "
echo
read USE?"Describe the purpose of this refresh: "
echo
read TIME?"List the timestamp of the refresh source: "

echo "\nThis instance is being used by $FNAME $LNAME for the purpose of:"
echo "$USE\n"
echo "This will result in a refresh of data as of: $TIME\n"

echo "\nOkay to proceed with the refresh? (y/n/q): \c"; read ANS
case $ANS in
    y|Y)  tput clear
          RUN_STAT=1
          echo "\n  `date +%H:%M`: Initiating refresh of $OSID using $BTYPE_TEXT backup.";;
    q|Q)  exit;;
     *)  echo "Cycle through setup screens again" > /dev/null
          RUN_STAT=0;;
esac
}
# End Function 'confirm_refresh'
################
```

```
################
#
function prep_db_files {

echo "  `date +%H:%M`: Removing current contents of staging area."
rm $STAGE/* > /dev/null 2>&1
rm $REDO/* > /dev/null 2>&1

echo "  `date +%H:%M`: rcp'ing $COPY_DIR backup to staging area.\t\t\t(be patient)"
rcp -p cagln1s111:/nearline/backup/PROD/${COPY_DIR}/d*/oradata/PROD/* ${STAGE}
chown -R oracle:dba ${STAGE}

if [ $BU_TYPE_FLAG -eq 1 ]; then
  echo "  `date +%H:%M`: rcp'ing $COPY_DIR redo to staging area."
  rcp -p cagln1s111:/nearline/backup/PROD/${COPY_DIR}/redo/* ${REDO}
  uncompress ${REDO}/*
  chown -R oracle:dba ${REDO}
fi

#
# Remove control files...
#
rm $STAGE/cntrl* > /dev/null 2>&1

}
# End Function 'prep_db_files'
################


################
#
function oracle_control {

case $1 in
  STOP)
#       if [ -f $ORACLE_HOME/dbs/sgadef${OSID}.dbf ]; then
# Obsolete in 8i
        if [ `ps -ef | grep pmon_${OSID} | wc -l` -gt 1 ]; then
          echo "  `date +%H:%M`: Stopping ConcMgrs."
          /usr/lbin/conc_mgr_control stop $OSID > /dev/null 2>&1
          echo "  `date +%H:%M`: Stopping Database."
          $BBIN/db_control.ksh stop $OSID > /dev/null 2>&1
        else
          echo "  `date +%H:%M`: Database and ConcMgrs already shutdown."
        fi

        echo "  `date +%H:%M`: Clearing application and database logs/reports."
# Test for 11i - database only - don't touch the Log/Output directories
        if ! [[ $APPL_TOP = "" ]]; then
          find $APPLCSF/$APPLOUT -type f -exec rm -f {} \; > /dev/null 2>&1
        fi
        rm -f ${OBASE}/${OSID}/bdump/* > /dev/null 2>&1
```

```
        rm -f ${OBASE}/${OSID}/cdump/* > /dev/null 2>&1
        rm -f ${OBASE}/${OSID}/udump/* > /dev/null 2>&1
  ;;
  START)
        echo "  `date +%H:%M`: ConcMgrs can be started now (theoretically)."
        echo "  `date +%H:%M`: Starting ConcMgrs."
        /usr/lbin/conc_mgr_control start $OSID > /dev/null 2>&1
  ;;
  *)
        echo "\n  ERROR: oracle_control function improperly called!\n"
        exit
  ;;
esac

}
# End Function 'oracle_control'
################


################
#
function do_code_refresh {

if [ $APPL_TOP != "/" ] && [ $APPL_TOP != "" ]; then
  cd $APPL_TOP > /dev/null 2>&1

  if [ $? -ne 0 ]; then
    echo "\n  ERROR: Cannot change directory to $APPL_TOP"
    echo   "         to perform code refresh!\n"
    exit
  fi

  echo "  `date +%H:%M`: Clearing application code ($APPL_TOP)."
  rm -rf ./* > /dev/null 2>&1
  echo "  `date +%H:%M`: Extracting application code tree.\t\t\t(be patient)"
  zcat $CODE_TAR | tar xvf - > /dev/null
  rm -rf ./install/log/* > /dev/null 2>&1
  rm -f ./*.env > /dev/null 2>&1
  cp -p ./../*.env . > /dev/null 2>&1
  chown -R applmgr:appls ./* /dev/null 2>&1
  chmod -R 777 ./fci/srw ./fci/sql /dev/null 2>&1
else
  echo "\n  ERROR: \$APPL_TOP variable not set!  Cannot proceed with code refresh!\n"
  exit
fi

}
# End Function 'do_code_refresh'
################


################
#
```

```
function do_db_refresh_1 {

MAP_FLAG=0
MISS_FILES=""
OUTPUT=uncomp${ORACLE_SID}.ksh

if [ $BU_TYPE_FLAG -eq 0 ]; then
  RECR_SQL="$OBASE/$ORACLE_SID/create/recr${ORACLE_SID}.sql"
elif [ $BU_TYPE_FLAG -eq 1 ]; then
  RECR_SQL="$OBASE/$ORACLE_SID/create/recr${ORACLE_SID}hot.sql"
fi

if [ ! -f $RECR_SQL ]; then
  echo "\n  ERROR: Cannot locate recreate script:"
  echo   "         $RECR_SQL\n"
  exit
fi

#
#  Check to see if any files mapped in the recreate script do not exist
#  in the staging directory...
#
for TFILE in `grep oradata $RECR_SQL | grep -v redo | grep -v temp | grep -v cntrl | tr -d ",';" `
do
  RECR_FILE="${STAGE}/`basename $TFILE`.Z"
  if [ ! -f $RECR_FILE ]; then
    MAP_FLAG=1
    MISS_FILES="${MISS_FILES} `basename $TFILE`"
  fi
done

if [ $MAP_FLAG -ne 0 ]; then
  echo "\n  ERROR: File(s) mapped in the recreate script do not exist in"
  echo   "         the staging directory!"
  echo "\n         Recreate Script:  ${RECR_SQL}"
  echo   "         Missing File(s): ${MISS_FILES}"
  echo "\n         Correct problem and re-run this program.\n"
  exit
fi

echo "  `date +%H:%M`: Building sample serial uncompress script."

for i in `ls $STAGE/*`
do
  FILE=`basename $i | sed -e 's/\.Z//'`
  if [[ ${FILE} != *stby_control.dbf ]]; then
  FCNT=`grep /$FILE $RECR_SQL | wc -l`
  if [[ $FCNT -eq 0 ]]; then
    if [[ ${FILE} = *control* || ${FILE} = *cntrl* ]]; then
    echo "uncompress -c $i > /d01/oradata/$ORACLE_SID/${FILE}"
    else
    echo "uncompress -c $i > /dXX/oradata/$ORACLE_SID/$FILE"
    fi
```

```
  else
    if [[ ${ORACLE_SID} = PROD && ${FILE} = temp* ]]; then
    UFILE=`grep /$FILE ${RECR_SQL} |awk '{print $6}' |sed s/\'//g |sed s/,//`
    echo "uncompress -c $i > ${UFILE}"
    elif [[ ${FILE} = redo* ]]; then
    UFILE=`grep /$FILE ${RECR_SQL} |awk '{print $3}' |sed s/\'//g |sed s/,//`
    echo "uncompress -c $i > ${UFILE}"
    else
    UFILE=`grep /$FILE ${RECR_SQL} |awk '{print $1}' |sed s/\'//g |sed s/,//`
    echo "uncompress -c $i > ${UFILE}"
    fi
  fi
  fi
done | sort +3 > $OUTPUT

#
#  Check to see if all database files are mapped within the
#  recreate script...
#
if [ `grep /dXX $OUTPUT | wc -l` -eq 0 ]; then
  echo "  `date +%H:%M`: Removing current datafiles."
  rm /d*/oradata/$ORACLE_SID/* > /dev/null 2>&1

#  chmod 755 $OUTPUT
#  ./$OUTPUT
#  chown oracle:dba /d*/oradata/${ORACLE_SID}/*

else
  echo "\n  ERROR: One or more files in staging area not mapped in:"
  echo   "         `pwd`/${OUTPUT}"
  echo   "         Search for the pattern 'dXX', modify the file as needed,"
  echo   "         and re-run this program.\n"
  exit
fi
}
# End Function 'do_db_refresh_1'
#*******************************************************************************
#
# Begin func_do_restore function
#
#*******************************************************************************
function func_do_restore {

echo "  `date +%H:%M`: Running uncompress script with $PARALLEL worker(s).\t\t(be patient)"
#
# This loop creates $PARALLEL variables prefixed with "BU_GV"
# to drive the compress processes...
#
 F_CNT=1
 for F in `ls $STAGE/*.Z`
   do
     eval "BU_GV${F_CNT}=\"\$BU_GV${F_CNT} $F\""
```

```
    if [ $F_CNT = $PARALLEL ]; then
      F_CNT=1
    else
      ((F_CNT=F_CNT+1))
    fi
  done

#
# Perform uncompress of backup to disk...
#
echo "  `date +%H:%M`: Datafile restoration portion began."

# Stack parallel processes by parsing envvars array BU_GV(1..n)
F_CNT=1
PROCS=""
while [ $F_CNT -le $PARALLEL ];
  do
    PROCS="${PROCS} BU_GV${F_CNT}"
    ((F_CNT=F_CNT+1))
  done

  for P in $PROCS
    do
      eval "func_df_restore \${P} &"
    done

wait

chown oracle:dba /d*/oradata/${ORACLE_SID}/*
echo "\n  `date +%H:%M`: Datafile restore portion ended."

}
# End func_do_restore function

#*******************************************************************************
#
# Begin func_df_restore function
#
#*******************************************************************************
function func_df_restore {

FILES="$*"

for i in $FILES
  do
    FILE=`basename $i | sed s/.Z//`
    if [[ ${FILE} = *control* || ${FILE} = *cntrl* ]]; then
    UFILE=/d01/oradata/${ORACLE_SID}/${FILE}
    elif [[ ${ORACLE_SID} = PROD && ${FILE} = temp* ]]; then
    UFILE=`grep /$FILE ${RECR_SQL} |awk '{print $6}' |sed s/\'//g |sed s/,//`
    elif [[ ${FILE} = redo* ]]; then
    UFILE=`grep /$FILE ${RECR_SQL} |awk '{print $3}' |sed s/\'//g |sed s/,//`
    else
```

```
   UFILE=`grep /$FILE ${RECR_SQL} |awk '{print $1}' |sed s/\'//g |sed s/,//`
   fi
   uncompress -c $i > ${UFILE}
done

}
# End func_df_restore function
#*******************************************************************************


################
#
function do_db_refresh_2 {

echo "  `date +%H:%M`: Calling db recreate script: ${OBASE}/bin/refresh_slave01.ksh"
echo "         (Output captured in ${OBASE}/bin/${ORACLE_SID}_slave01.log)"
${OBASE}/bin/refresh_slave01.ksh ${ORACLE_SID} ${OBASE} ${RECR_SQL} \
  > ${OBASE}/bin/${ORACLE_SID}_slave01.log 2>&1

if [ $PW_RESET_FLAG = 1 ]; then
  . /usr/lbin/${OSID}_setup
  echo $SYSTEM_PW | sqlplus system @${OBASE}/bin/update_fnduser.sql > /dev/null
  typeset -l SIDLC=${OSID}
  echo "  `date +%H:%M`: Performing automatic password reset...\n"
  rsh cagln1s91 ". /usr/lbin/${OSID}_setup; /oracle/admin/passupdt/passchg.ksh"
  echo "  `date +%H:%M`: Executing AD Auto Configure (respond with APPS pw when
prompted...\n"
  rsh cagln1s91 ". /usr/lbin/${OSID}_setup; /oracle/${SIDLC}util/adautocfg.sh"
  echo "  `date +%H:%M`: Auto Configure of $OSID complete.\n"
fi

}
# End Function 'do_db_refresh_2'
################


################
#
function do_post_refresh_notice {
echo "  `date +%H:%M`: Sending e-mail refresh complete notice.\n"

SUBJECT="Refresh of $OSID database is complete."
MESSAGE=/usr/tmp/.refresh_message
DATABASE="v\$database"
#
# Set FinSys email list.
#
FINSYS_EMAIL="me@me.com"

USERS_EMAIL="users@me.com"

#
# Get current date of refresh.
```

```
#
RDATE=`sqlplus -s apps/$APPS_PW << EOF
SET    HEAD  off
SET    FEED  off
SET    ECHO  off
SET    PAGES 0
SELECT   created
FROM     $DATABASE;
exit
EOF`

RTIME=`sqlplus -s apps/$APPS_PW << EOF2
SET    HEAD  off
SET    FEED  off
SET    ECHO  off
SET    PAGES 0
SELECT   MAX(start_time)
FROM     fnd_logins;
exit
EOF2`

#
# Generate message.
#
echo "\nThe refresh of the $OSID database has been completed as of $RDATE" > $MESSAGE
echo "\nThis instance is being used by $FNAME $LNAME for the purpose of:" >> $MESSAGE
echo "$USE" >> $MESSAGE
echo "\nThis refresh has been performed using data as of: $RTIME" >> $MESSAGE
echo "\nPlease contact this person prior to other use of this instance. \n" >> $MESSAGE

#
# Send message.
#
 cat $MESSAGE | mailx -s "$SUBJECT" $FINSYS_EMAIL $USERS_EMAIL
 rm $MESSAGE


}
# End Function 'do_post_refresh_notice'
################

################
#
function do_db_test {
# Oracle 8i compatibility - JHL
# if [ ! -f ${ORACLE_HOME}/dbs/sgadef${ORACLE_SID}.dbf  ]; then
if [ `ps -ef | grep pmon_${ORACLE_SID} | wc -l` -lt 2 ]; then
  # Test 1:  check whether database is running.
  REFRESH_OK=0
else
  # Instance is okay; file exists.
  REFRESH_OK=1
fi
```

```
if [ $REFRESH_OK -eq 1 ]; then
  echo "  `date +%H:%M`: Refresh of $OSID complete.\n"
  do_post_refresh_notice
else
  echo "  `date +%H:%M`: Refresh of $OSID complete did not complete"
  echo "  `date +%H:%M`: successfully. Please check on current status and"
  echo "  `date +%H:%M`: re-execute the refresh as necessary.\n"
fi

}
# End Function 'do_db_test'
################

#*****************************************************************  #
#                                                                    #
#              P R O G R A M    B O D Y                 #
#                                                                    #
#*****************************************************************  #


RUN_STAT=0

while [ $RUN_STAT -eq 0 ]
do
  set_pw_reset_flag
  set_code_flag

  . /usr/lbin/fnd_setup
  . /usr/lbin/ora_pass

  if [[ $ORACLE_SID = "" && $TWO_TASK != "" ]]; then
    OSID=$TWO_TASK
    ORACLE_SID=$TWO_TASK
    VERFLAG="8_true"
  elif [[ $ORACLE_SID != "" ]]; then
    OSID=$ORACLE_SID
    VERFLAG="7_true"
  else
    echo "\nORACLE_SID not set successfully - cannot continue."
    exit
  fi

  if [[ $ORACLE_SID = "PBDEV1" ]]; then
     echo "Cannot be used to refresh Paybase at this time!\n"
     exit
  fi

  set_backup_source

  confirm_refresh
done
```

```
oracle_control STOP

if [ $STAGE_FLAG -eq 0 ]; then
  prep_db_files
fi

if [ $CODE_FLAG = 1 ]; then
  do_code_refresh
fi

do_db_refresh_1

func_do_restore

do_db_refresh_2

oracle_control START

do_db_test
```

Attachment 3 *global_updates.sql*

```
--
--
-- Reset system and sys passwords
--
--
ALTER USER system IDENTIFIED BY manager;
ALTER USER sys IDENTIFIED BY change_on_install;
--
--
-- Drop Discoverer Tablespace and user accounts
--
--
-- drop user case cascade;
-- drop user case31 cascade;
-- drop tablespace case including contents;
--
--
-- Change all pending requests to 'Completed', 'Cancelled' except for
-- 'Purge Concurrent Requests'
--
--
UPDATE applsys.fnd_concurrent_requests
SET    phase_code = 'C', status_code = 'D'
WHERE  phase_code != 'C'
AND    concurrent_program_id != 32263;
--
--
-- Decrease number of concurrent processes
--
--
UPDATE applsys.fnd_concurrent_queue_size a
SET    min_processes = 1
WHERE EXISTS
  ( SELECT 'x'
    FROM   applsys.fnd_concurrent_queues b
    WHERE  a.concurrent_queue_id = b.concurrent_queue_id
    AND    b.concurrent_queue_name = 'STANDARD')
/
UPDATE applsys.fnd_concurrent_queue_size a
SET    min_processes = 1
WHERE EXISTS
  ( SELECT 'x'
    FROM   applsys.fnd_concurrent_queues b
    WHERE  a.concurrent_queue_id = b.concurrent_queue_id
    AND    b.concurrent_queue_name = 'FNDCRM')
/
UPDATE applsys.fnd_concurrent_queue_size a
SET    min_processes = 1
WHERE EXISTS
  ( SELECT 'x'
    FROM   applsys.fnd_concurrent_queues b
    WHERE  a.concurrent_queue_id = b.concurrent_queue_id
```

```
   AND    b.concurrent_queue_name = 'FNDLIBR236')
/
UPDATE applsys.fnd_concurrent_queue_size a
SET    min_processes = 1
WHERE EXISTS
  ( SELECT 'x'
    FROM   applsys.fnd_concurrent_queues b
    WHERE  a.concurrent_queue_id = b.concurrent_queue_id
    AND    b.concurrent_queue_name = 'FNDLIBR276')
/
UPDATE applsys.fnd_concurrent_queue_size a
SET    min_processes = 0
WHERE EXISTS
  ( SELECT 'x'
    FROM   applsys.fnd_concurrent_queues b
    WHERE  a.concurrent_queue_id = b.concurrent_queue_id
    AND    b.concurrent_queue_name = 'FNDSM_CAGLN1S92')
/
--
--
-- Enable FSG Development Responsibility for all users
--
--
UPDATE    applsys.fnd_user_resp_groups a
SET       end_date = NULL
WHERE EXISTS
  ( SELECT 'x'
    FROM   applsys.fnd_responsibility_tl b
    WHERE  b.responsibility_name = 'EIG General Ledger FSG GUI'
    AND    a.responsibility_application_id = b.application_id
    AND    a.responsibility_id = b.responsibility_id )
/
--
--
-- Enable Sysadmin Responsibilities for all FinSys Users
--   1469 = JLUI

--
UPDATE    applsys.fnd_user_resp_groups a
SET       end_date = NULL
WHERE EXISTS
  ( SELECT 'x'
    FROM   applsys.fnd_responsibility_tl b
    WHERE  b.responsibility_name like '%System Admin%'
    AND    a.user_id in (1469)
    AND    a.responsibility_application_id = b.application_id
    AND    a.responsibility_id = b.responsibility_id )
/
--
--
-- Enable all 'GUI' Responsibilities (except SysAdmin) for all finsys users
--
--
```

```
-- 1/12/98 - Added UIDs for SMARQUAR, SALLEN (1867, 1106)
-- 10/14/99 - Add FGC users - JL
--   4556 = RJOSEPH
--   4559 = EVYSTOSK
UPDATE    applsys.fnd_user_resp_groups a
SET       end_date = NULL
WHERE EXISTS
  ( SELECT 'x'
    FROM   applsys.fnd_responsibility_tl b
    WHERE  b.responsibility_name like '%GUI'
    AND    b.responsibility_name not like '%System Admin%'
    AND    a.user_id in (1027,1038,1039,1107,1469,
                         2072,2171,6670,6793,7199)
    AND    a.responsibility_application_id = b.application_id
    AND    a.responsibility_id = b.responsibility_id )
/
--
-- SQL to allow document forwarding, self-approval,
-- appover to modify and altering approval path for all
-- document types:
-- C. Light - 2/15/99
--
UPDATE  po.po_document_types_all
SET     can_change_forward_from_flag='Y',
        can_preparer_approve_flag='Y',
        can_approver_modify_doc_flag = 'Y',
        can_change_approval_path_flag ='Y',
        can_change_forward_to_flag='Y'
/
--
--  SQL to reactivate Buyer status for active employees:
--  C. Light - 2/15/99
--
UPDATE  po.po_agents
SET     end_date_active = NULL
WHERE   end_date_active IS NOT NULL
AND     agent_id IN
        (SELECT person_id
         FROM apps.per_people_f
         WHERE TO_CHAR(effective_end_date)='31-DEC-12'
               AND person_id in
               (SELECT person_id
                FROM apps.per_assignments_f
                WHERE TO_CHAR(effective_end_date)='31-DEC-12'
               )
        )
/
--
-- SQL to reactivate Position Controls (approval authority)
-- for all document types if an active employee holds the position:
-- C. Light - 2/15/99
--
UPDATE  po.po_position_controls_all
```

```
SET     end_date = NULL
WHERE   end_date IS NOT NULL
AND     position_id IN
       (SELECT position_id
        FROM apps.per_positions
        WHERE position_id IN
             (SELECT position_id
              FROM apps.per_assignments_f
              WHERE TO_CHAR(effective_end_date)='31-DEC-12'
              AND person_id IN
                   (SELECT person_id
                    FROM apps.per_people_f
                    WHERE TO_CHAR(effective_end_date) = '31-DEC-12')
                   )
             )
/
--
--
-- Enable Banner Page printing
--
--
UPDATE    applsys.fnd_printer_drivers fpd
SET       arguments = 'banner.ksh $PROFILES$.PRINTER $PROFILES$.CONC_COPIES
$PROFILES$.TITLE $PROFILES$.FILENAME'
WHERE EXISTS
  ( SELECT   'x'
    FROM     applsys.fnd_printer_information fpi
    WHERE    fpi.printer_type = 'HPLJ4SI'
    AND      fpd.printer_driver_name = fpi.printer_driver
    AND      fpd.printer_driver_name <> 'FCI_RPT_SPLIT')
/
--
--
-- Drop apps triggers to stop annoying alerts whilst resetting passwords.
--
--
-- drop trigger apps.alr_fnd_user_responsibilit_iar
-- /
-- drop trigger apps.alr_fnd_user_responsibilit_uar
-- /
-- drop trigger apps.alr_fnd_user_iar
--- /
-- drop trigger apps.alr_fnd_user_uar
-- /
--
--
-- Drop apps triggers to stop self-posted journal checking.
--
--
drop trigger apps.alr_gl_je_headers_iar
/
drop trigger apps.alr_gl_je_headers_uar
/
```

```
--
--
-- Disable the Vista trigger
--
--
-- alter trigger apps.qsvp_build_print_list disable;
--
--
-- Disable all Object*Migrator functionality...
-- Added 19-MAR-1999, Lee Turner
--
--
UPDATE  applsys.fnd_concurrent_programs cp1
SET     cp1.enabled_flag = 'N'
WHERE   cp1.concurrent_program_id IN (
        SELECT cp.concurrent_program_id
        FROM   applsys.fnd_concurrent_programs cp, applsys.fnd_application fa
        WHERE  fa.application_short_name = 'CLM'
        AND    cp.application_id = fa.application_id )
/
UPDATE  applsys.fnd_responsibility a
SET     a.end_date = SYSDATE
WHERE   EXISTS
  (SELECT 'x'
   FROM applsys.fnd_responsibility_tl b
   WHERE b.responsibility_name LIKE 'EIG Object*Migrator%'
   AND   a.application_id = b.application_id
   AND   a.responsibility_id = b.responsibility_id );
/
COMMIT;
```

Attachment 4 *conc_mgr_control*

```
#!/bin/ksh
#
#  Modifications:  09/25/2000 - TEST and TST2 support for WFMAIL notification
#                             mailer.
#
#                 03/09/2003 - TEST and MOVM are 11i - Beta testing middle
#                             tier control method using rsh
#
#  Databases:
#
#  PRE    = Pre-Production
#  PRE2   = Pre-Production 2
#  MDD    = Staging
#  STG2   = Staging 2
#  TEST   = Test
#  TST2   = Test 2
#  FSG    = FSG Development
#  MOVM   = Move/Merge
#  RUTH   = FGC Instance
#  D107   = Training / Demo
#
if [ $# -eq 0 ]; then
  echo "\nUsage: $0 <start/stop> {SID}\n"
  echo "       If no {SID} is provided, starts/stops all instances.\n"
  exit
fi
. /usr/lbin/ora_pass
umask 002
typeset -l CONTROL=$1
shift

if [ $# -eq 0 ]; then
  SID_LIST="PRE STG2 MOVM TEST TST2"
else
  SID_LIST=$*
fi

case $CONTROL in

stop)
   for SID in $SID_LIST
     do
     echo "\nStopping concurrent managers for $SID database...\n"

     . /usr/lbin/${SID}_setup

       case $SID in
       D107)
            touch $APPLCSF/$APPLLOG/shutdown
            CONCSUB apps_appdemo/$APPS_PW SYSADMIN 'System Administrator' \
```

```
            SYSADMIN WAIT=Y CONCURRENT FND ABORT
                  # WAIT=Y CONCURRENT FND DEACTIVATE
        ;;
        MOVM|TEST|TST2|PRE|EICN)
            typeset -l SIDLC=${SID}
            rsh -n -l oracle cagln1s91 ". /usr/lbin/${SID}_setup; \
             /oracle/${SIDLC}comn/admin/scripts/${SID}/adstpall.sh \
             apps/$APPS_PW"
            if [[ ${SID} != "TEST" ]]; then
            rsh -n -l wf${SIDLC} cagln1s91 ". /usr/lbin/${SID}_setup; \
             /oracle/${SIDLC}appl/eigc/11.5.0/bin/fci_nm_control stop" &
            wait
            fi
            echo "$SID stopped.\n"
        ;;
        *)
            touch $APPLCSF/$APPLLOG/shutdown
            CONCSUB apps/$APPS_PW SYSADMIN 'System Administrator' SYSADMIN \
            WAIT=Y CONCURRENT FND ABORT
           # WAIT=Y CONCURRENT FND DEACTIVATE
        ;;
        esac
      done
    ;;

start)
    for SID in $SID_LIST
      do
        echo "\nStarting concurrent managers for $SID database...\n"

        . /usr/lbin/${SID}_setup

        case $SID in
          D107)
            rm $APPLCSF/$APPLLOG/shutdown
            startmgr sysmgr=apps_appdemo/$APPS_PW
          ;;
          MOVM|TEST|TST2|PRE)
            typeset -l SIDLC=${SID}
            LOGFILE="/oracle/rshout.tmp"
            rsh -n -l oracle cagln1s91 ". /usr/lbin/${SID}_setup; .
/oracle/${SIDLC}comn/admin/scripts/${SID}/adstrtal.sh apps/$APPS_PW"
            if [[ ${SID} != "TEST" ]]; then
            rsh -n -l wf${SIDLC} cagln1s91 ". /usr/lbin/${SID}_setup; .
/oracle/${SIDLC}appl/eigc/11.5.0/bin/fci_nm_control start > /dev/null 2>&1 " &
            fi
            echo "$SID started.\n"
          ;;
          *)
            rm $APPLCSF/$APPLLOG/shutdown
            startmgr sysmgr=apps/$APPS_PW
          ;;
        esac
```

```
    done
  ;;

*)
  echo "\n  ERROR: Usage: conc_mgr_control [start|stop] [sid list]\n"
  ;;

esac
```

Attachment 5 *recr<SID>hot.sql*

```
set echo on;
spool recrPRE.lst

connect / as sysdba;

STARTUP NOMOUNT;
CREATE CONTROLFILE REUSE
SET DATABASE PRE
    MAXLOGFILES 16
    MAXLOGMEMBERS 2
    MAXDATAFILES 1022
    MAXINSTANCES 1
    MAXLOGHISTORY 100
LOGFILE
  GROUP 1 '/d01/oradata/PRE/redo01.dbf'  SIZE 20M,
  GROUP 2 '/d01/oradata/PRE/redo02.dbf'  SIZE 20M,
  GROUP 3 '/d01/oradata/PRE/redo03.dbf'  SIZE 20M,
  GROUP 4 '/d01/oradata/PRE/redo04.dbf'  SIZE 20M,
  GROUP 5 '/d01/oradata/PRE/redo05.dbf'  SIZE 20M RESETLOGS
/* STANDBY LOGFILE
   GROUP 6 '/d01/oradata/PRE/redosb01.dbf'  SIZE 20M,
   GROUP 7 '/d01/oradata/PRE/redosb02.dbf'  SIZE 20M,
   GROUP 8 '/d01/oradata/PRE/redosb03.dbf'  SIZE 20M,
   GROUP 9 '/d01/oradata/PRE/redosb04.dbf'  SIZE 20M,
   GROUP 10 '/d01/oradata/PRE/redosb05.dbf'  SIZE 20M */
DATAFILE
  '/d01/oradata/PRE/system01.dbf',
  '/d01/oradata/PRE/system02.dbf',
  '/d01/oradata/PRE/system03.dbf',
  '/d01/oradata/PRE/apd01.dbf',
<...snip...>
  '/d01/oradata/PRE/xtrx01.dbf',
  '/d01/oradata/PRE/portal01.dbf'
;

ALTER DATABASE RECOVER AUTOMATIC DATABASE
UNTIL TIME '2005-01-29:01:51:00'
USING BACKUP CONTROLFILE;

ALTER DATABASE OPEN RESETLOGS;

ALTER TABLESPACE TEMP ADD TEMPFILE '/d01/oradata/PRE/temp01.dbf' SIZE 2000M
REUSE;

UPDATE sys.global_name
   SET global_name = 'PRE.WORLD';

UPDATE applsys.fnd_product_groups
   SET release_name = '11.5.8',
       applications_system_name = 'PRE';
```

```
UPDATE applsys.fnd_profile_option_values v
   SET v.profile_option_value = 'PRE Development Database'
 WHERE EXISTS (SELECT 'x'
                 FROM applsys.fnd_profile_options o
                WHERE o.profile_option_name = 'SITENAME'
                  AND o.profile_option_id = v.profile_option_id);

@/oracle/admin/bin/global_updates.sql

exit;
```

Attachment 6 *update_fnduser.sql*

```
--
--
-- Expire all application users' passwords to force change at first login.
--
--
UPDATE applsys.fnd_user
SET    password_date = NULL
WHERE  user_name NOT IN ('SYSADMIN','GUEST','PORTAL30','PORTAL30_SSO');
--
--
-- Disable all application users' accounts except for those listed.
-- Ignore if FSG instance.
--
--
UPDATE applsys.fnd_user
SET    end_date = sysdate
WHERE user_name NOT IN ('SYSADMIN','INTERFACE','GUEST',
               'PORTAL30','PORTAL30_SSO','ITPROJECTS',
               'MVERDIGU','CKIMN','MMOYER','JLUI','RWAITE',
               'GELLIS','TMARLOW','DGARRETT','CLO','MPRUCH',
               'SLEE','DBURT','HBANJARY','RSHINN','VGALVAN',
               'MRAMELLI','CMCCOOL','CEVANS','VDIZON')
AND NOT EXISTS (SELECT 'x'
             FROM   v$database
             WHERE  name = 'FSG');
commit;
```

Attachment 7 *db_control.ksh*

```
#!/bin/ksh
###############################################################################
#
#
#  filename:  db_control.ksh
#     usage:  db_control.ksh [start|stop] <SID list>
#        by:  Lee Turner
#      date:  Oct 13, 1997
#
#   updated:  10/22/98, Lee Turner
#             Changed from if statement to case statement and changed
#             first parameter to start|stop.
#
#            09/07/2000 JHL
#              Modified to support Server Partitioning with 8i
#              Note: the original <SID>_setup files must accommodate
#              placing the standard ORACLE_HOME pointing at the 7.3.4
#              file locations.  <SID>D_setup files have been created to
#              allow the database utilities to use the 8.1.x home without
#              causing error messages (due to the differences in required
#              support files between the two versions).
#
#      desc:  This script is called by the metiocb.ksh (cold) backup script
#             and is used to either shutdown or startup the database
#             being backed up.  Normally, this code is included in the
#             main backup script as the script is run by root with SUID
#             as oracle.  However, Sequent PTX does not support SUID
#             for shell scripts so this code must be externalized and
#             called from the main script via a 'su -c oracle' command.
#
###############################################################################
#

typeset -l CONTROL=$1
shift

if [ $# -eq 0 ]; then
  SID_LIST="PROD"
else
  SID_LIST=$*
fi

for X in $SID_LIST
  do
    if [ `grep $X /etc/oratab | wc -l` -eq 0 ]; then
      echo "\n  ERROR: Invalid ORACLE_SID: '$X'\n"
      exit
    fi
  done

case $CONTROL in
```

```
stop)
   for SID in $SID_LIST
     do
       . /usr/lbin/${SID}D_setup

sqlplus /nolog << EOC
connect / as sysdba;
shutdown immediate;
exit;
EOC
 done
   ;;

start)
   for SID in $SID_LIST
     do
       . /usr/lbin/${SID}D_setup
sqlplus /nolog << EOC1
connect / as sysdba;
startup;
exit;
EOC1
     done
   ;;

*)
   echo "\n  ERROR: Usage: db_control.ksh [start|stop] [sid list]\n"
   ;;

esac
```

Attachment 8 *refresh_slave01.ksh*

```
#!/bin/ksh
#
# Called by /oracle/bin/refresh_master.ksh
#

ORACLE_SID=$1
OBASE=$2
RECR_SQL=$3

. /usr/lbin/${ORACLE_SID}D_setup
. /usr/lbin/ora_pass
#
# Recreate orapwd file...
#
echo "  `date +%H:%M`: Recreating orapwd file (first stage)."
rm $ORACLE_HOME/dbs/orapw$ORACLE_SID
orapwd file=$ORACLE_HOME/dbs/orapw$ORACLE_SID password=$PROD_SYSTEM_PW

#
# Recreate controlfile/database; global_updates.sql should be called from
# recr${ORACLE_SID}.sql script...
#
echo "  `date +%H:%M`: Bootstrapping $ORACLE_SID database."
sqlplus /nolog << EOF
@$RECR_SQL
EOF

#
# Recreate orapwd file - final stage...
#
echo "  `date +%H:%M`: Recreating orapwd file (final stage)."
rm $ORACLE_HOME/dbs/orapw$ORACLE_SID
orapwd file=$ORACLE_HOME/dbs/orapw$ORACLE_SID password=$SYSTEM_PW

#
# Recreate OEM schema – Obsolete with 9i Databases
#
echo "  `date +%H:%M`: Recreating OEM schema."
sqlplus /nolog << EOF2
@$OBASE/bin/recrOEM.sql
EOF2
```

Attachment 9 *passchg.ksh*

```
#!/bin/ksh
# passchg.ksh - create the 11i Applications password change routine
#               scripts.
# J. Lui - 6/4/2003 - 1st inception.
# Set MODE to REAL for non Development Testing
MODE="REAL"
if [[ $ORACLE_SID = "" ]]; then
  ORASID=${TWO_TASK}
else
  ORASID=${ORACLE_SID}
fi
LDIR=/oracle/admin/passupdt
BINNAME=$LDIR/passupdt.ksh
SQL1NAME=$LDIR/passupdt_pt1.sql
SQL2NAME=$LDIR/passupdt_pt2.sql
LOGFILE=$LDIR/passupdt.log
cat /dev/null > $LOGFILE
echo "  `date +%H:%M`: Changing passwords to defaults \
for $ORASID - $MODE mode." | tee -a $LOGFILE
. /usr/lbin/ora_pass
#
# Change SYSTEM and SYS passwords back to defaults
#
if [[ $MODE = "REAL" ]]; then
  sqlplus system/$SYSTEM_PROD_PW @$SQL1NAME $SYSTEM_PW $SYS_PW
else
  # Development to Development Test mode
  sqlplus system/$SYSTEM_PW @$SQL1NAME $SYSTEM_PW $SYS_PW
fi
echo "\n  `date +%H:%M`: SYSTEM and SYS user passwords changed to defaults \
for $ORASID\n" | tee -a $LOGFILE
#
# Generate FNDCPASS shell script
#
if [[ $MODE = "REAL" ]]; then
 sqlplus apps/$APPS_PROD_PW @$SQL2NAME $BINNAME $APPS_PROD_PW \
 $APPS_PW $SYSTEM_PW
else
  # Development to Development Test mode
  sqlplus apps/$APPS_PW @$SQL2NAME $BINNAME $APPS_PW \
       $APPS_PW $SYSTEM_PW
fi
echo "\n  `date +%H:%M`: Application User password update script \
created - ${BINNAME}\n" | tee -a $LOGFILE
#
# Change Applications passwords back to defaults
#
echo "\n  `date +%H:%M`: Executing... ${BINNAME}" | tee -a $LOGFILE
. $BINNAME > /dev/null
cat ${BINNAME} >> $LOGFILE
if [[ -f $LDIR/aferror.log ]]; then
```

```
  cat -s $LDIR/aferror.log >> $LOGFILE
  rm -f $LDIR/aferror.log
fi
cat -s $LDIR/L*.log >> $LOGFILE
rm -f $LDIR/L*.log
echo "\n  `date +%H:%M`: Log file $LOGFILE generated." | tee -a $LOGFILE
echo "  `date +%H:%M`: Checking for Errors...."
grep rror $LOGFILE | tee -a $LOGFILE
echo "  `date +%H:%M`: $ORASID updated back to default \
passwords. \n" | tee -a $LOGFILE
#
```

Attachment 10 *passupdt_pt1.sql*

```
set newp 1
set hea off
set feed off
set verify off
set term off
set echo off
set pages 200
--
-- Reset system and sys passwords
--
--
ALTER USER system IDENTIFIED BY &1
/
ALTER USER sys IDENTIFIED BY &2
/
exit
```

Attachment 11 *passupdt_pt2.sql*

```
set newp 1
set hea off
set feed off
set verify off
set term off
set echo off
set pages 200
--
-- Create Change script to submit FNDCPASS at OS level
--
spool &1
select 'FNDCPASS apps/&2 0 Y system/&4 SYSTEM APPLSYS &3'
from    DUAL
/
select  'FNDCPASS apps/&3 0 Y system/&4 ORACLE '||
        ORACLE_USERNAME || ' ' || ORACLE_USERNAME || '4'
from    FND_ORACLE_USERID
where   READ_ONLY_FLAG='A'
/
exit
```