Oracle Migration Plan – Detailed Description

The following document adds some detail to the steps outlined in the Oracle Migration Plan Spreadsheet. The spreadsheet is a step by step, high level, outline of what must be done to migrate the current Solaris 9i database to Windows Azure running 11g.

**Note:** While the steps in the spreadsheet are numbered, those numbers are not referred to here as it is possible for the spreadsheet to be updated.

**NOTE: All** database work must be carried out with the following settings in the Windows cmd session:

set oracle\_sid=<database being worked on>

set oracle\_home=C:\OracleDatabase\product\11.2.0\dbhome\_1

set nls\_lang=american\_america.we8iso8859p1

set nls\_date\_format=

or

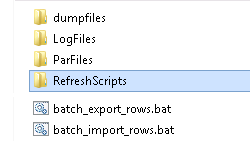
c:\utilities\oraenv <database being worked on>

set nls\_date\_format=

***The NLS\_DATE\_FORMAT must be unset or the NOROWS import will fail due to a badly defined column default value which uses a date format that is not necessarily the one you may wish to define.***

# Important Note

**All scripts, parameter files etc are assumed to be located in a tree structure similar to the following:**

****

* **DumpFiles** Holds the uncompressed export files and logs from the production database.
* **LogFiles** is where the import jobs will write their logfiles.
* **ParFiles** is where the import jobs read their parameter files from.
* **RefreshScripts** is where the various scripts used in the refresh are to be found.
* **Scripts** (not shown above) is where the common scripts and output from the production reconciliation scripts etc. Also here are scripts that must be generated by the production DBAs and are required to be run on the migrated database after the migration has completed.
* There may be other folders present, but these are not used.

The image above shows the view, as it were, from the root directory**. It is from this root directory that everything in the following document takes place.**

# Production DBA Tasks

* **Ensure receipt** of most recent copies of the scripts to generate the package that creates the export parameter files. These can change from time to time and the latest version must be used. The supporting documentation must also be supplied. The scripts are in TFS at location $/TA/DEV/Projects/Oracle Upgrade 9i to 11g/UKRegulated/Database/DBA Documentation/Code/ExportParfileScripts.
* **Ensure receipt** of the current database reconciliation scripts. Again, these might be changed and the latest version must be used. The scripts are in TFS at location $/TA/DEV/Projects/Oracle Upgrade 9i to 11g/UKRegulated/Database/DBA Documentation/Code/ITReconcilliationScripts.
* **Create the export parameter files**:
  + exp\_NOFCS
  + exp\_FCS1
  + exp\_FCS2D
  + exp\_FCS3
  + exp\_FCS4
  + exp\_FCS5
  + exp\_FCS6
  + exp\_FCS7
  + exp\_FCS8
  + exp\_FCS9
  + exp\_NOROWS
* The production DBAs should stop all processes that connect to the production database prior to the exports being run.
* Run script *S095\_RecreatePUBLICDatabaseLinks.sql* to generate a script, *T095\_RecreatePublic\_DB\_Links.sql*, which rebuilds the PUBLIC database links. This is run as part of the *PreRefresh* script. The output needs to be sent to the Leeds DBAs.
* Run the reconciliation scripts, and supply the results to the Leeds DBAs, before exporting the database. These scripts are documented in "*UV Database Migration Guide*" which can be found in TFS at location $/TA/DEV/Projects/Oracle Upgrade 9i to 11g/UKRegulated/Database/Upgrade Documentation.
* Export the database using in above parameter files. Run these in parallel to reduce the overall run time.
* As each export completes, check the logfile for problems and fix as necessary. If there are no problems, zip up the dump file and SFTP to the appropriate Azure server. Please also SFTP the logfile.

# Leeds DBA Tasks

* **Ensure all required *'RefreshScripts'* scripts have been copied from TFS, to the server**. The scripts are in TFS at location $/TA/DEV/Projects/Oracle Upgrade 9i to 11g/UKRegulated/Database/DBA Documentation/Code/migrationRootFolder. These should be checked out and copied to the appropriate database server in Azure.
* Also, **check out and copy the 'Scripts'** folder, from the same TFS location, to the database server.
* Copy the supplied reconciliation scripts output from the Production DBAs, into the 'Scripts' folder, overwriting anything that is already there.
* Make sure that any/all previous runs/test/etc have had their log files etc removed from the root of the migration folder, and the logfiles folder.
* **Make sure** that the NOROWS, NOROWS\_GRANTS and NOFCS import parameter files are edited to ensure that **full=y** has been used instead or **FROMUSER**.
* **Disable Data Guard, if running.** This is unlikely to be necessary. However, if it is required to do so, then the steps are outlined in the document "*AZURE – Using RMAN to create standby databases.docx*" which is in TFS at location $/TA/DEV/Projects/Oracle Upgrade 9i to 11g/UKRegulated/Database/DBA Documentation/RMAN Cloning to Standby. The appropriate section is entitled **Stopping Managed Recovery**.
* **If you wish**, before starting SQL\*Plus for the following scripts, save yourself having to type in "RefreshScripts\" or "Scripts\" all the time by running the following 'set SQLPATH' command, replacing '?' by the appropriate drive letter.

set SQLPATH=?:\Migrationroot\refreshscripts;?:\Migrationroot\scripts

* Change to the MigrationRoot folder:

cd ?:\MigrationRoot

* **Edit the script** (supplied by the production DBAs) *Scripts\T095\_RecreatePublic\_DB\_Links.*sql to remove the CFGTEST database link.
* **Edit the script** supplied by the production DBAs, *Scripts\T150a\_create\_roles.sql*. Look for "**alter user fcs**" (near line 1550) and check that the entire command is on one line – because the script that generated it didn't set lines wide enough, it has wrapped - The text 'AD\_ROLE,COMMS\_ROLE,WEB\_USER' may be on a second line, and this breaks the SQL. Make it a single line if necessary.
* Start SQL\*Plus as SYSDBA

oraenv <database\_name>

set NLS\_DATE\_FORMAT=

sqlplus sys/password as sysdba

* **Run the ShutDownRestart script.** The script *RefreshScripts\ShutDownRestart.sql* must be run in order to prepare the database for the migration. It will:
  + Shutdown and MOUNT the database;
  + Disable flashback mode;
  + Disable Archivelog mode;
  + Open the database for use.
* **Run the preRefresh script.** There is a script named *RefreshScripts\preRefresh.sql* which must be executed using SQL\*Plus before any other work is carried out on the Azure server. This script will delete and recreate the desired users and set up any required initial grants and privileges to those (new) users.
* **Check logfiles created:**
  + *create\_profiles.lst*
  + *PreRefresh.lst*
  + *Grants.lst*
  + *DropPublicDBLinks.lst*
  + *T095\_recreatePublic\_DB\_links.lst*
  + *Drop\_old\_users.lst*
  + *Drop\_old\_users\_2.lst*
  + *create\_users\_and\_roles.lst*
  + *create\_tablespace\_quotas.lst*
  + *create\_system\_privs.lst*
  + *create\_proxies.lst*
  + *create\_roles.lst*
* The Production DBAs will have placed zipped copies of all the export files onto the SFTP location on the production servers. **Unzip the export files on Azure.** On the Azure servers, the export files must be placed in the "*DumpFiles*" folder previously created for this purpose. The files should be unzipped using 7Zip as opposed to WinZip as 7Zip is by far the faster of the two utilities.
* **Check the logfiles for any errors.** It is possible that the production DBAs overlooked one or more errors in the export. As a sanity check, it is advisable to scan the files for any errors. This can be automated to a degree by running the following command:

find /I "EXP-" \*.log

Any errors will be listed along with the logfile that the error was found in.

* **Import NOROWS dumpfile.** This file recreates the empty structure of the database accounts. It is imported by running the following command:

set nls\_date\_format=

imp parfile=parfiles\imp\_NOROWS.par

When complete, check the logfile, as above, for any errors. These must be repaired before continuing.

find /i "imp-" logfiles\imp\_NOROWS.log | find /v /i "00041" | find /v "encountered"

* Error IMP-00015 for public synonym PRODUCT\_IMPL can be ignored.
* Error 2270 for XML\_FATCA\_REPORTS can be ignored.
* Error 12014 for CREATE SNAPSHOT LOG on INVESTOR and ORDTRAN can be ignored.
* Error 6564 for any table named 'EXT\_%' can be ignored as directory THREAD\_EXT\_TABLES is no longer created. (These EXT tables are later deleted anyway.)
* **Run the post\_Import\_norows script.** Using SQL\*Plus, connect as the SYS user and execute the *RefreshScripts\post\_import\_norows.sql* script to:
  + Issue required grants;
  + Recompile some invalid objects;
  + Disable various triggers;@shutdownrestart
  + Configure some database parameters prior to the main imports.
  + Drop some XML stuff that breaks the imports;
  + Drops some packages that audit the imports and cause them to take much longer;
  + Drops 9i specific Materialized Views and snapshots;
  + Drops the existing DBMS\_JOBS for the FCS user.
* **Run the various ROWS imports.** A script has been supplied to execute the required parallel imports - *batch\_import\_rows.bat* – see below *before* executing it. Please note that FCS9 will not be imported as it always fails. FCS8 will usually fail at the last table too. These are resolved below.

**Note**: The userid is, by necessity, hard coded into the various parameter files as the FCS user. If this is unsuitable, or if the live weekend has required that the FCS password is different from that in the development Azure environment, then *all* the parameter files should be edited to suit.

**The script *must* be executed in a DOS session, and is named *batch\_import\_rows.bat*. It will take care of submitting all the required imports. If you only double-click the file in Explorer, then everything will appear and vanish in a flash, *and will not work*.**

However, if something happens and one or more imports fail to run correctly, you may *'type batch\_import\_rows.cmd*' to view the appropriate commands and rerun the appropriate one for the failed import.

**Using a Toad Session Browser, make sure that none of the import sessions end up waiting for a "*SQL\*Net message from client*" for a long period of time, a few seconds is fine, but longer may indicate that that session's DOS window has gone into SELECT mode and will prevent the import from writing to the screen. This will hang the import session.**

**Because the imports were started with the DOS START command, the title bars do not show "select" when they are in this mode, unlike ordinary DOS sessions.**

**If a session does enter this mode, click its window, and hit the RETURN key a few times until refreshing Toad shows that the session is no longer waiting.**

**The FEEDBACK parameter in the import parameter files *may* cause this as the import needs to print a dot every so often to show rows are being imported. If the screen is in SELECT mode, it will hang. For this reason, this parameter has been removed from the parameter files following a nasty 17 hour hang with no indications as to why.**

You can use the following script to get a pretty good idea of how fast things are progressing. It may barf with a divide by zero error if a table hasn't imported any rows yet, just run it again after a couple of seconds if this results. (It will work, soon! Keep trying.)

-- How fast is my import runnig?

-- BEWARE, sessions that are creating indexes

-- will show a decreasing "rows per minute" figure as there

-- are no more rows importing, but time is still passing!

--

set lines 2000 pages 2000 trimspool on

col table\_name for a31

col index\_name for a31

select substr(sql\_text,instr(sql\_text,'INTO "') +6,instr(sql\_text, '(') - instr(sql\_text,'INTO "') -8) table\_name,

null index\_name,

rows\_processed,

round((sysdate-to\_date(first\_load\_time,'yyyy-mm-dd hh24:mi:ss'))\*24\*60,1) minutes,

trunc(rows\_processed/((sysdate-to\_date(first\_load\_time,'yyyy-mm-dd hh24:mi:ss'))\*24\*60)) rows\_per\_min

from sys.v\_$sqlarea

where sql\_text like 'INSERT %INTO "%'

and command\_type = 2

and open\_versions > 0

--

union all

--

select replace(substr(sql\_text,instr(sql\_text,'ON "') +4,instr(sql\_text, '(') - instr(sql\_text,'ON "') -6),'"', null) table\_name,

replace(substr(sql\_text,instr(sql\_text,'INDEX "') +7,instr(sql\_text, ' ON') - instr(sql\_text,'INDEX "') -8),'"', null) index\_name,

null rows\_processed,

round((sysdate-to\_date(first\_load\_time,'yyyy-mm-dd hh24:mi:ss'))\*24\*60,1) minutes,

null rows\_per\_min

from sys.v\_$sqlarea

where sql\_text like 'CREATE %INDEX%'

and command\_type = 9

and open\_versions > 0

--

-- List the table first, then the index creation, if any.

order by 1, 2 nulls first;

* **Potential Problem 1:** There are two XML tables which rely on a cascade of different XML types. One is in **FCS8** ~~and the other in~~ **~~FCS9~~**. (FCS9 is never run now.) These types have an internal OID (Object ID) and on the import, these are recreated so the tables subsequently refuse to import as they "require" a different OID for the various Types. In addition, one table has a mixed case name.

The Oracle workaround for this problem doesn't appear to work.

**In general, if one of these two tables fails to import, there will be problems with the other. See below for the fixes.**

In the event of any XML problems in **FCS8**~~/~~**~~FCS9~~**, the fixes, which are detailed below, should be run *after* the end of the *NOROWS\_GRANTS* import, and *before* the start of the *post\_import\_rows* scripts.

* **Potential Problem 2:** FCS6 *might* fail to create index ALERT\_LOG\_PK due to ALERT\_LOG\_SEQ duplicates *somehow* created by the PK\_ALERTS.RUN\_HEARBEAT scheduled job. If this happens, find the duplicates as follows:

select alert\_log\_seq,count(\*)

from fcs.alert\_log

group by alert\_log\_seq

having count(\*) > 1

order by alert\_log\_seq;

Then, for each duplicate alert\_log\_seq listed, find the details and the ROWID, as follows:

select rowid, alert\_log\_seq, message

from fcs.alert\_log

where alert\_log\_seq in ( whatever you got above)

order by alert\_log\_seq, message;

Then, after choosing the duplicate you want to delete, delete using the ROWID which is the quickest manner of deleting a row, or two, from a massive table with no indexes:

delete from fcs.alert\_log

where rowid in (chartorowid('xxx'), chartorowid('yyy'), ...);

Where 'xxx' and 'yyy' are the desired ROWIDs that you wish to delete. When the SQL completes and you have confirmed that the number of rows deleted is what you expected, commit the changes:

commit;

exit

And finally, run the following import to fixup the indexes that failed to create:

imp parfile=parfiles\temp\_fcs6.par

* **Potential Problem 3:** FCS5 may fail to create indexes due to error "*ORA-01555 Snapshot too old*". This will happen on slow servers, or fast ones with the speed turned right down! The log file will contain the failing CREATE INDEX commands, so extract , tidy up the double quotes, and re-execute in SQL\*Plus while connected as FCS, not SYS. One or more of the following will probably be needed:

CREATE INDEX AUIT\_LOG\_IX2 ON AUDIT\_LOG (AUDITUSER )

PCTFREE 10 INITRANS 2 MAXTRANS 255

STORAGE(INITIAL 65536 FREELISTS 1 FREELIST GROUPS 1)

TABLESPACE CFGLOG\_INDEX LOGGING;

CREATE UNIQUE INDEX AUDIT\_LOG\_PK ON AUDIT\_LOG (AUDITID )

PCTFREE 10 INITRANS 2 MAXTRANS 255

STORAGE(INITIAL 65536 FREELISTS 1 FREELIST GROUPS 1)

TABLESPACE CFGLOG\_INDEX LOGGING;

CREATE INDEX AUDIT\_LOG\_IX3

ON AUDIT\_LOG (PRIMARYKEY , PRIMARYKEYNAME , AUDITTABLE )

PCTFREE 10 INITRANS 2 MAXTRANS 255

STORAGE(INITIAL 65536 FREELISTS 1 FREELIST GROUPS 1)

TABLESPACE CFGLOG\_INDEX LOGGING;

CREATE INDEX PSO\_AUDIT\_LOG\_NX01 ON AUDIT\_LOG (AUDITDATE )

PCTFREE 10 INITRANS 2 MAXTRANS 255

STORAGE(INITIAL 65536 FREELISTS 1 FREELIST GROUPS 1)

TABLESPACE UVDATA01\_INDEX LOGGING;

* **Check the logfiles for any errors.** This can be automated to a degree by running the following commands. The first just shows the exit status of each import, the seconds filters errors we need to be concerned about.

find /i "Import terminated" logfiles\\*.log

find /i "IMP-" logfiles\imp\_rows\*.log | find /i /v "1917"

* Execute the script *RefreshScripts\drop\_fcs\_jobs.sql* as we don't want errors in the NOROWS\_GRANTS step which follows. It may fail, but this is ok – the jobs may not be present.
* **Run the NOROWS\_grants import.** The tables have been imported and the data etc are all present. At this stage some of the grants have been set up by the *RefreshScripts\grants.sql* script, however, any new tables or procedures etc will not have had their grants included in that script. In addition to granting required permissions, this import also:
  + Creates the various constraints required;
  + Re-creates the recently deleted jobs owned by FCS as scheduler jobs;
  + Re-creates the two packages TABLE\_AUDIT and PK\_ALERTS dropped above;
  + Recompiles all PL/SQL;
  + Recompiles all triggers and enables them.

Run the following command to carry out the above:

start "GRANTS" /d . /high imp parfile=parfiles\imp\_NOROWS\_grants.par

The following script will assist in monitoring progress in the absence of Toad:

set lines 2000 pages 2000 trimspool on

col sql\_text for a100

select sql\_id, sql\_text

from v$sql

where sql\_id = (

select nvl(sql\_id, prev\_sql\_id)

from v$session

where program = 'imp.exe'

);

* **Check the logfile for any errors.** This can be automated to a degree by running the following command:

find /i "IMP-" logfiles\imp\_norows\_grants.log | find /I /v "1917" | find /i /v "different identifier" | find /v "error 1:"| find /v "encountered"

You can ignore errors relating to the constraint FATCA\_FILE\_SUBMISSION\_FK01 on table fatca\_file\_submission as it references XML\_FATCA\_REPORTS which we will recreate below.

* **Re-import the XML tables** this will only be required in there were errors with the XML tables in **FCS8** ~~and/or~~ **~~FCS9~~**:

sqlplus sys/<password> as sysdba

@RefreshScripts\drop\_xml\_stuff

exit

imp parfile=parfiles\temp\_fcs9.par

sqlplus fcs/<password>

@RefreshScripts\fix\_xml\_stuff

* Check the log, *fix\_xml\_stuff.lst* for errors.
* **Run the post import script.** The script is named *RefreshScripts\post\_import\_rows.sql.* It will run for a fair length of time as it has quite a lot of work to do, including (but not limited to) the following:
  + Issuing a lot more grants;
  + Gathering database statistics;
  + Creating various constraints;
  + Enabling table logging;
  + Recompiling any remaining invalid objects;
  + ~~Configuring the correct password on the roles NORMAL\_USER and SVC\_AURA\_SERV\_ROLE;~~
  + Rebuild the two Materialised Views – FCS.INVESTOR\_CAT\_MV and FCS.ORDTRAN\_MV.
  + Building the new UVSCHEDULER\_ROLE;
  + Granting new system privileges to the SVC\_AURA\_SERV user.

Progress can be checked with the following script if Toad is not available. While the gathering of stats is executing, yo can see the current table with this query:

set pages 2000 lines 2000 trimspool on

select action from v$session where module = 'UPGRADE: Gather Stats';

And the actual command in execution with the following:

set pages 2000 lines 2000 trimspool on

select sql\_id, sql\_text

from v$sql

where sql\_id = (

select nvl(sql\_id, prev\_sql\_id)

from v$session

where program = 'sqlplus.exe'

and sid <> (select sid from v$mystat where rownum = 1)

--and sql\_text like 'begin dbms\_stats%'

)

* Check for errors with the command:

find /i "ORA-" post\_import\_rows.lst | find /v "01951" | find /v "01921"

There may be ORA-00942 errors relating to '*FCS.SYS\_%==*' tables. These relate to various TYPEs that have been created for XML, LOB out of line storage etc. If the table names are '*SYS\_%==*' then the error can be ignored, otherwise, fix it.

* **Run the following SQL** but *only* if this is a non-production database:

column db\_name new\_value my\_dbname noprint;

select name as db\_name from v$database;

alter role NORMAL\_USER identified by &&my\_dbname.123;

alter role SVC\_AURA\_SERV\_ROLE identified by &&my\_dbname.123;

alter system set service\_names='&&my\_dbname' scope=both;

alter system set instance\_name='&&my\_dbname' scope=spfile;

* Run the script *Scripts\T170\_Create\_Public\_Synonyms.sql* to recreate all public synonyms. Errors here can simply be ignored – there are a number, around 32, of invalid public synonyms in the production database at the time of writing.
* **Run the postRefresh script.** Only for pre-production and production imports. The script is named *RefreshScripts\postRefresh.sql* and it will:
  + Shutdown and MOUNT the database;
  + Enable ARCHIVELOG mode;
  + Enable FLASHBACK mode;
  + Open the database for use.
* **Check XML table.** Table *FCS."UKFATCASubmissionFIRe98\_TAB"* - yes, it is in mixed case - may report that the table "has errors" when you:

SELECT \* FROM FCS."UKFATCASubmissionFIRe98\_TAB";

"No rows selected" is expected, if any errors result, then drop the table and recreate it as follows:

DROP TABLE FCS."UKFATCASubmissionFIRe98\_TAB" CASCADE CONSTRAINTS PURGE;

CREATE TABLE fcs."UKFATCASubmissionFIRe98\_TAB" OF "XMLTYPE"

XMLSCHEMA "http://hmrc.gov.uk/UKFATCASubmissionFIReport" ELEMENT "UKFATCASubmissionFIReport"

PCTFREE 10

PCTUSED 40

INITRANS 1

MAXTRANS 255 NOCOMPRESS LOGGING STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

PCTINCREASE 0

FREELISTS 1

FREELIST GROUPS 1

BUFFER\_POOL DEFAULT

)

TABLESPACE UVDATA01;

SELECT \* FROM FCS."UKFATCASubmissionFIRe98\_TAB";

* **Run the reconciliation scripts.** Execute script *RefreshScripts\run\_reconcilliation\_scripts.sql*.
* **Check that the results** match, or are better, than those supplied by the production DBAs. For the object count comparison, it may be best to utilise a 'diff' utility such as WinMerge or similar to make the checks. Any mention of the *FCS.DEPERSONALISATION* package and any tables named *DEPERS%* script can be ignored. The results scripts are located in the *Scripts* folder.
* For all databases. If not previously done, create a PERFSTAT tablespace (2gb, autoextend, unlimited, next 50m) and run the scripts to create & install the PERFSTAT user. Normally, this will have been carried out at database creation, but just in case:

@'?\rdbms\admin\spcreate'

exit -- From PERFSTAT user.

* For production databases only. Also, if creating the user above, create the auto jobs to take snapshots and purge old ones:

connect SYS/<password> as sysdba

grant create job to perfstat;

connect perfstat/<password>

@RefreshScripts\PERFSTAT\_AUTOJOB.sql

@RefreshScripts\PERFSTAT\_AUTOPURGE.sql

exit -- From PERFSTAT user.

* **Execute the script** *RefreshScripts\CheckSystemObjects.sql* to determine if any non-system users have their default tablespace set to SYSTEM, and if so, do they have any objects in the SYSTEM tablespace. If anything appears ("no rows selected" is the desired outcome here) then:
  + **Execute the script** *RefreshScripts\MoveSystemObjects.sql* to create SQL commands to move the affected objects out of SYSTEM into CFA, which is the new default tablespace we are using for the affected users.
  + **Save any output from the above script** as SQL commands to move the affected objects out of SYSTEM. They need to be run later, once the default tablespace has been set and quota allocated to these users. **NOTE:** in testing, no users had quota on system so no objects were possible.
* **Run the script** *RefreshScripts\MoveDefaultTablespace.sql* to correct those users who have SYSTEM as their default tablespace. Check the logfile, *MoveDefaultTablespace.lst* (in the current directory) for errors.
* If there were any objects needing moved listed above by *RefreshScripts\MoveSystemObjects.sql* then the owners of those objects will need quota on CFA. **Manually grant appropriate quota** - unlimited is perhaps not the best option! Check in DBA\_TS\_QUOTAS to see what they have currently on SYSTEM and give that on CFA. Make sure that you also grant a zero quota on SYSTEM.
* If there were any objects needing moved listed above by *RefreshScripts\MoveSystemObjects.sql* then the generated commands can be executed to move objects out of SYSTEM.
* Run the script *RefreshScripts\Create\_FCS\_Scheduler\_Jobs.sql* to convert the current set of DBMS\_JOBs for FCS into 11g DBMS\_SCHEDULER jobs instead as the former is no longer used (from 10g). **The jobs will be created disabled and will therefore not run.**
* For the Live weekend only - run the following SQL to enable required jobs, **while connected as the FCS user account**:

begin

dbms\_scheduler.enable(name => 'ALERTS\_HEARTBEAT');

dbms\_scheduler.enable(name => 'CLEARLOGS');

dbms\_scheduler.enable(name => 'JISA\_18BDAY\_CONVERSION');

end;

/

At this point, it is appropriate for the users to carry out any testing that is necessary *after* any services etc have been pointed to the new database, and activated. This is out with the scope of the DBA Team.

* For the Live weekend only - run the following SQL to create replacements for the Solaris Cronjobs, **while connected as the SYS user account**:

@RefreshScripts\Solaris\_cronjobs.pks

@RefreshScripts\Solaris\_cronjobs.pkb

@RefreshScripts\Scheduler\_jobs.sql

* **NOTE**: On the production and pre-production servers there are Windows Task Scheduler jobs created to run RMAN backups daily for the CFG, CFGAUDIT and CFGRMN databases. These have been set to disabled and will need to be enabled at this point but only on the primary servers uvorc01 and ppduvorc01.

On the standby servers, uvorc02, ppduvorc02 and druvorc03, these tasks are disabled by default and should only be enabled when there is a switchover or failover.

While the user testing is ongoing, all of the following tasks can be carried out in parallel with testing.

* **Create a standby database**. The standby database should now be created as per the document "*AZURE – Using RMAN to create standby databases.docx*" which is in TFS at location $/TA/DEV/Projects/Oracle Upgrade 9i to 11g/UKRegulated/Database/DBA Documentation/RMAN Cloning to Standby.
* **Configure RMAN Backups**.
* **Backup the database**. The newly migrated database should have a backup taken. A cold backup is preferred however, **be aware** that this will require the database to be shutdown and MOUNTed, which will affect any testing that is ongoing. Consider taking an online backup if testing is still in progress.

Backups scripts are available in c:\utilities\RMAN on the server, or, in TFS at location $/TA/DEV/Projects/Oracle Upgrade 9i to 11g/UKRegulated/Database/DBA Documentation/Code/BackupScripts. These can be used to take a cold or hot backup of the database using RMAN.

The following tasks are specific to DevOps, but not to the DBA Team. They constitute the user testing part of the migration (mentioned above) and can be run in parallel at any stage from the actual existence of the database.

* Repoint monitoring to the newly migrated database.
* Repoint all services and items on the "onion" diagram at the newly imported database.

The following tasks are also specific to DevOps, and not to the DBA Team. They constitute more of the user testing part of the migration (mentioned above) and can be run in parallel at any stage from after the database has been imported and the post import scripts executed to completion.

* Test connectivity for all services and/or "onion" diagram component.
* Enable/switch on each service.
* Begin user specific application testing.

# Post Weekend Tasks

The following tasks are considered to be required after a successful migration. They are all outwith the scope of this document however.

* Switch off the 9i database.
* Carry out a warranty period for the migrated database.
* Decommission the 9i database server.
* Repurpose the existing (9i?) Oracle licence(s).