Oracle Banking Trade Finance

Multi-Tenant Patch-set Deployment

Release 14.7.5.0.0
G15576-01
September 2024

ORACLE

Oracle Banking Trade Finance Multi-Tenant Patch-set Deployment, Release 14.7.5.0.0

G15576-01

Primary Authors: (primary author), (primary author)

Contributing Authors: (contributing author), (contributing author)

Contributors: (contributor), (contributor)

Contents

1

Overview of Applications in an Application Container

1.1 Managing Applications in an Application Container

In an application container, an application is the named, versioned set of application common objects stored in the application root. In this context, “application” means “application back-end.” Application common objects include user accounts, tables, PL/SQL packages, and so on. An application can be shared with the application PDBs that belong to the application root.

On performing application changes, application PDBs can synchronize with the application in the application root. The application container also manages the versions of the application and the patches to the application:

- While installing an application, user must specify the application version number.
- While upgrading an application, user must specify the old application version number and the new application version number.

As the application evolves, the application container maintains all of the versions that are applied.

1.2 Application Maintenance

Application maintenance refers to installing, uninstalling, upgrading, or patching an application.

Perform application installation, upgrade, and patching operations using an ALTER PLUGGABLE DATABASE APPLICATION statement.

The basic steps for application maintenance are as follows:

1. Log in to the application root.
2. Begin the operation with an ALTER PLUGGABLE DATABASE APPLICATION ... BEGIN statement in the application root.
3. Execute the application maintenance statements.
4. End the operation with an ALTER PLUGGABLE DATABASE APPLICATION ... END statement.

These statements can be issued in the same user session or in different user sessions.

- Application Installation
- Application Upgrade
  An application upgrade is a major change to an installed application.

1.2.1 Application Installation

An application installation is the initial creation of a master application definition. A typical installation creates user accounts, tables, and PL/SQL packages.

Refer **Multi-Tenant_Deployment.pdf** for more details on the application installation.

1.2.2 Application Upgrade

An application upgrade is a major change to an installed application.

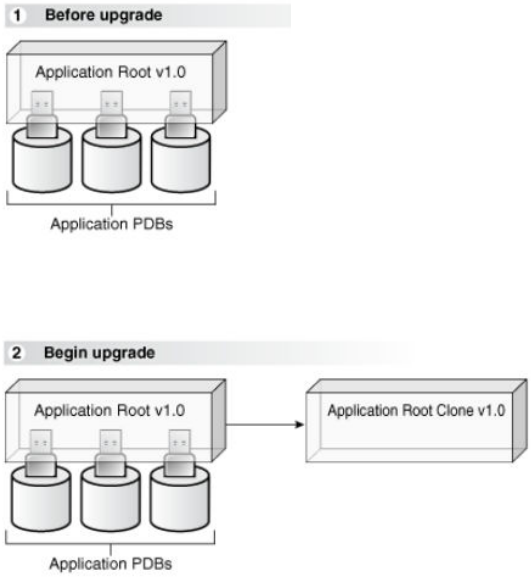
Typically, an upgrade changes the physical architecture of the application. For example, an upgrade might add new tables, and packages, or alter the definitions of existing objects.

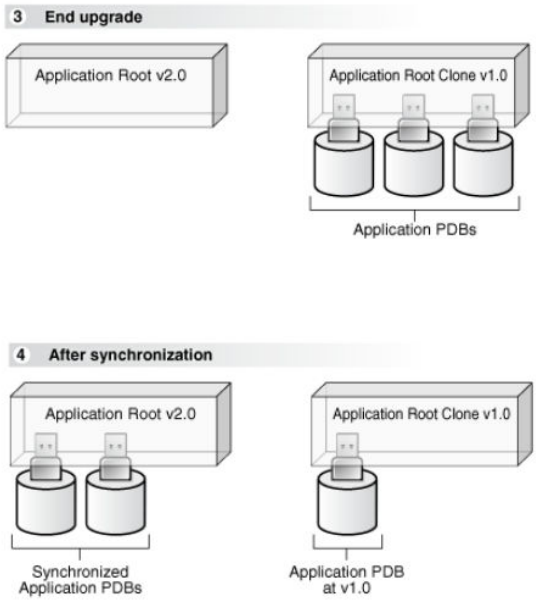
To upgrade the application, specify the following in the ALTER PLUGGABLE DATABASE APPLICATION statement:

- Name of the application
- Old application version number
- New application version number

During an application upgrade, the application remains available. To make this availability possible, Oracle Database clones the application root.

The following figure gives an overview of the application upgrade process.





When an application is upgraded, Oracle Database automatically clones the application root.

During the upgrade, application PDBs point to the clone and applications continue to run during the upgrade. Application PDBs can perform DML on metadata-linked and tables and views and query data-linked tables.

After the upgrade, the application root clone remains and continues to support any application PDB that still uses the pre-upgrade version of the application in the clone.

Application PDBs that re synchronized are pointed to the upgraded application root. Application PDBs that are not synchronized might continue to use the clone.

2
Patch-set Application Steps

Multi entity application root/PDB based setup has to be available to perform18c database application upgrade for applying the patch-set. Refer Multi-Tenant_Deployment.docx for the deployment and installation steps.

Patch-set can be applied by following below steps in sequential order, and detail of each steps explained as separate sections subsequently.

- Application Upgrade
- Synchronize application PDBs

Patch-set Deployment Pre-requisites:

- Download the required patch-set zip file and unzip it in a local path.
- Verify whether the property files (fcubs.properties and env.properties) have the application root schema details where the application is available, if not update the approot schema details through installer (Refer OBTF_Property_File_Creation.docx for more details) and re-generate the files.
- Make sure to set the flag PATCHSET_INSTALLATION to 'Y'.

2.1 Application Upgrade

2.1.1 Purpose

Major changes to an application constitute application upgrades. During the upgrade, Oracle Database automatically clones the application root and the application PDBs point to the clone.

Application upgrade can be performed in the application root only, and application PDBs applies the changes in the upgrade when they synchronize with the application.

2.1.2 Steps to be Followed

Below steps to be followed to initiate application upgrade:

- Start Application upgrade
- Compiling Incremental Units

- Recompilation of invalids
- End Application upgrade
- Start Application upgrade
- Application Root objects conversion for new objects
- Application Root objects conversion for existing objects
- Recompilation of invalids
- End Application upgrade
- Start Application Upgrade
  An ALTER PLUGGABLE DATABASE APPLICATION statement has to be issued to upgrade an application in the application root.
- Compiling Incremental Units
- Recompilation of invalids
- End Application upgrade
- Start Application upgrade
- Application Root Objects Conversion for New Objects
- Application Root Objects Conversion for Existing Objects
- Recompilation of Invalids
- End Application Upgrade

2.1.2.1 Start Application Upgrade

An ALTER PLUGGABLE DATABASE APPLICATION statement has to be issued to upgrade an application in the application root.

Each upgrade must be associated with an application name, starting version number, and ending version number.

- The common user must have the DBA privilege, and the privilege must be commonly granted in the application root.
- The application root must be in open read/write.
- Run the below script for initiating an application upgrade. This will initiate the application from current version to the next version (patch-set version).

  Start_Upgrade

  Input sample for the script:

Spool Path	<< Any local path>>
Application next version	14.4.0.0.0

2.1.2.2 Compiling Incremental Units

Patch-set objects have to be loaded using bat file [E.g.: SMSDBCompileRun.bat, TFDBCompileRun.bat] by silent installer for respective product processer.

Compile the incremental SMS units using `/INSTALLER/SOFT/SMSDBCompileRun.sh` for UNIX installations or `/INSTALLER/SOFT/SMSDBCompileRun.bat` for Windows installations.

Compile the incremental OBTF units using `/INSTALLER/SOFT/TFDBCompileRun.sh` for UNIX installations or `/INSTALLER/SOFT/TFDBCompileRun.bat` for Windows installations.

2.1.2.3 Recompilation of invalids

As the sharing property of most of the objects are modified other than NONE, recompilation of objects is not allowed outside an application.

Recompilation of objects will be initiated inside the application upgrade for sanity with zero invalids with the below script:

Recompilation of invalids

2.1.2.4 End Application upgrade

Application upgrade can be performed in the application root only and end of the upgrade is performed with an ALTER PLUGGABLE DATABASE APPLICATION END UPGRADE statement.

Run the below script for ending an application upgrade for patch-set.

End Application upgrade

And run the invalid script by connecting to the common user in approot outside the upgrade.

Invalid Recompilation Outside Upgrade.

2.1.2.5 Start Application upgrade

Run the below script for initiating another application upgrade for object conversion. This will initiate the application from current version to the next version (patch-set version).

Start Application upgrade

Input sample for the script:

Spool Path	<< Any local path>>
Application next version	14.4.0.0.0

2.1.2.6 Application Root Objects Conversion for New Objects

As part of patch-set when there are new tables added which has to be converted as DL or when there is a new function id which is identified to be an approot function is provided, otherwise no conversion will happen as part of this step

Below script takes care of converting the new DL objects during patch-set based on the deployment model of the application during installation.

New Object Conversion

Input sample for the script:

Spool Path	<< Any local path>>
Application next version	14.4.0.0.0

2.1.2.7 Application Root Objects Conversion for Existing Objects

Various Sharing types of objects during installation:

- A static table will hold the information of selected table sharing as Data link. Other tables will be treated as Meta Data Link
- Sharing of object types such as INDEX, LOB, TABLE PARTITION, SEQUENCE, and DYNAMIC PACKAGES will remain as NONE.
- All other object types such as Packages, Procedures, Functions, and Synonyms would be converted as Meta Data Link sharing.

Sharing during upgrade:
Sharing of existing database objects will remain the same.

Below script takes care of converting the modified MDL objects when there is a re-creation [objects with Create or Replace command during creation] happens during patch-set.
Application Root Objects Conversion

Input sample for the script:

Spool Path	<< Any local path>>
Application next version	14.4.0.0.0

When there are new tables introduced as part of patch-set which has to be converted into DL will be done separately. The recommendation for the same will be provided as part of patch-set instructions for this case.

2.1.2.8 Recompilation of Invalids

As the sharing property of most of the objects are modified other than NONE, recompilation of objects is not allowed outside an application.

Recompilation of objects will be initiated inside the application upgrade for sanity with zero invalids with the below script:

Recompilation of Invalids

2.1.2.9 End Application Upgrade

Application upgrade can be performed in the application root only and end of the upgrade is performed with an ALTER PLUGGABLE DATABASE APPLICATION END UPGRADE statement.

Run the below script for ending an application upgrade for patch-set.

End Application Upgrade

And run the invalid script by connecting to the common user in approot outside the upgrade.

Invalids_Recompilation_Outside_Upgrade

2.1.3 Purpose

Synchronizing an application updates the application in the application PDB to the latest version in the application root. When an application is upgraded in an application root, an application PDB that belongs to the application root is not changed until it is synchronized.

Application PDBs synchronize with an application by running an ALTER PLUGGABLE DATABASE statement with the SYNC clause.

2.1.4 Steps to be Followed

Prerequisites

- The current user must have ALTER PLUGGABLE DATABASE system privilege.
- Ensure that the current container is the application PDB.
- Run an ALTER PLUGGABLE DATABASE APPLICATION statement with the SYNC clause.
- Run the below script to synchronize the PDBs with the latest application changes in the application root.

  PDB_Sync

3
Step by Step Execution

3.1 Pre-Requisites

This topic provides systematic instructions for pre-requisites.

1. Before applying the patch-set, we have to make sure the release is updates with the base version of the patch-set.

   For Example, If the first patch-set of 14.2 is yet to applied, the release has to be updated as '14.2.0.0.0'. It can be verified with the below queries.

```
select param_name, param_val from CSTB_PARAM WHERE PARAM_NAME =
'RELEASE';
select module_group_id, release from SMTB_MODULES_GROUP;
```

2. Another significant parameter is the values of application name and deployment type in CSTB_PARAM.

   This value will be updated from the installer during Approot Object Conversion utility as part of deployment.

```
select param_name, param_val from cstb_param where PARAM_NAME in
('MULTI_TENANT_APP_NAME','MULTI_TENANT_DEPLOYMENT_MODEL');
```

   The Application name of multi-tenant deployment will be stored in CSTB_PARAM as

Param_Name	Param_Val
MULTI_TENANT_APP_NAME	OBTF

   The type of object conversion will be stored in CSTB_PARAM as

Param_Name	Param_Val
MULTI_TENANT_DEPLOYMENT_MODEL	SA (or) SAUA (or) SASDD (or) SASDC

   SA Shared Application
   SAUA Shared Application User Authentication
   SASDD Shared Application Shared Data - Default
   SASDC Shared Application Shared Data – Custom

(Optional) Enter the result of the procedure here.

3.2 Patch-set Application Step by Step with Screenshots

This topic provides systematic instructions to patch-set application step by step with screenshots.

1. ***Start Application upgrade***

   a. Login into the Approot Schema as Common user.

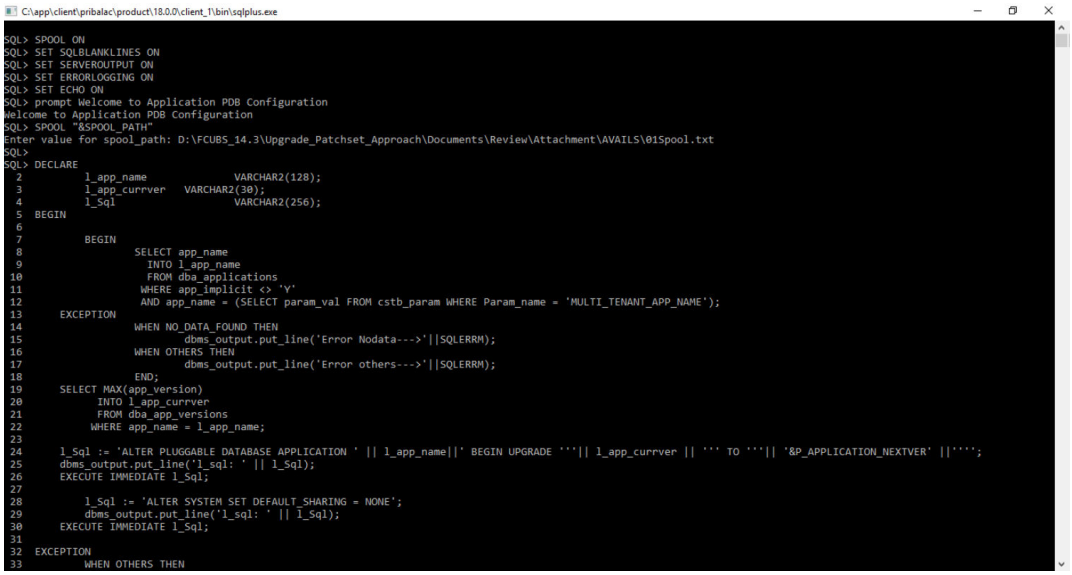
   b. Run `01_Start_Upgrade.sql` **for initiating the application upgrade**.

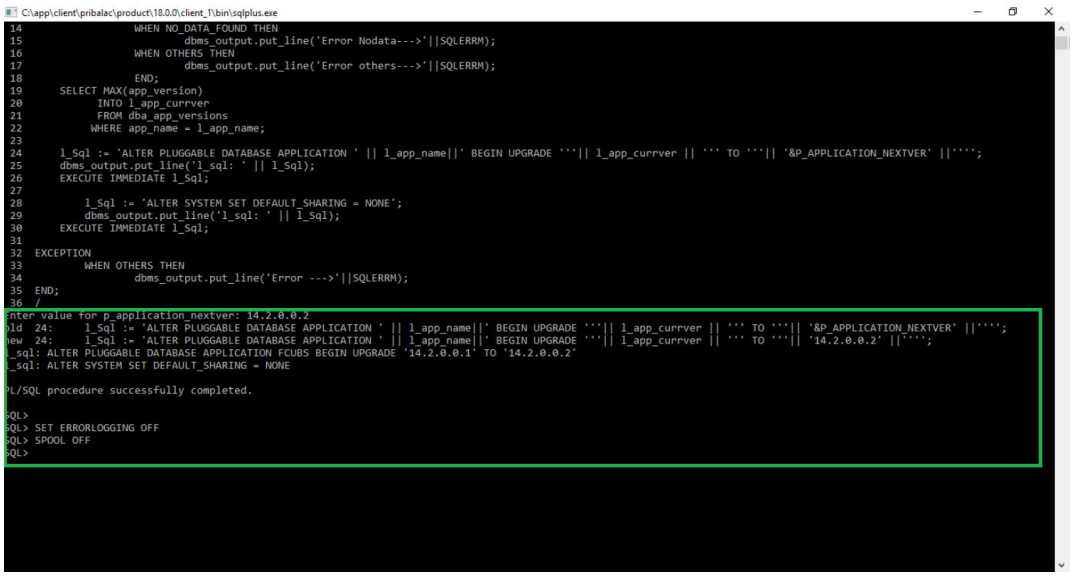
   c. User input has to be inputted for the below:

Spool Path	<< Any local path>>
Application next version	14.4.0.0.0

   d. Script will be executed as in the screen shot below and keep the SQL Plus session open for upcoming steps.

Execution Screenshot:





2. ***Compiling Incremental Units***

   a. Make sure that the fcubs.properties and env.properties are updated with approot schema details.

   b. Run the `<Product Processor>DBCompileRun.bat` from `<Patchset>\INSTALLER\SOFT` directory. DDL Compilation, Object Compilation and Static Data load will be done.

   For Example: OBTF INSTALLATION
   First load SMS objects first and then OBTF objects. i.e. Run `SMSDBCompileRun.bat` and after SMS object loading is completed, then initiate OBTF compilation Run `TFDBCompileRun.bat`

3. ***Recompilation of invalids***

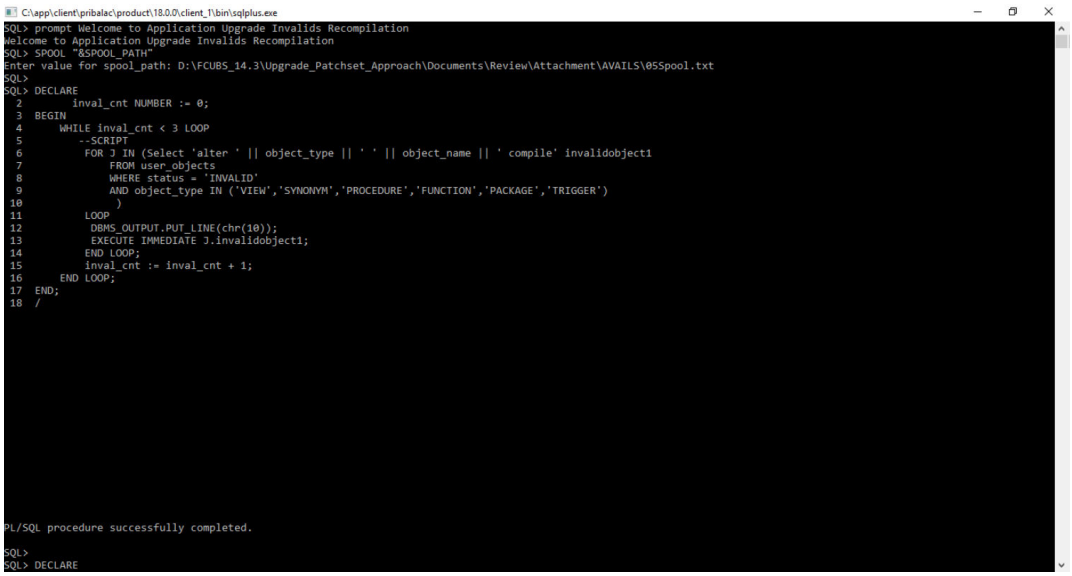
   a. Login into the Approot Schema as Common user

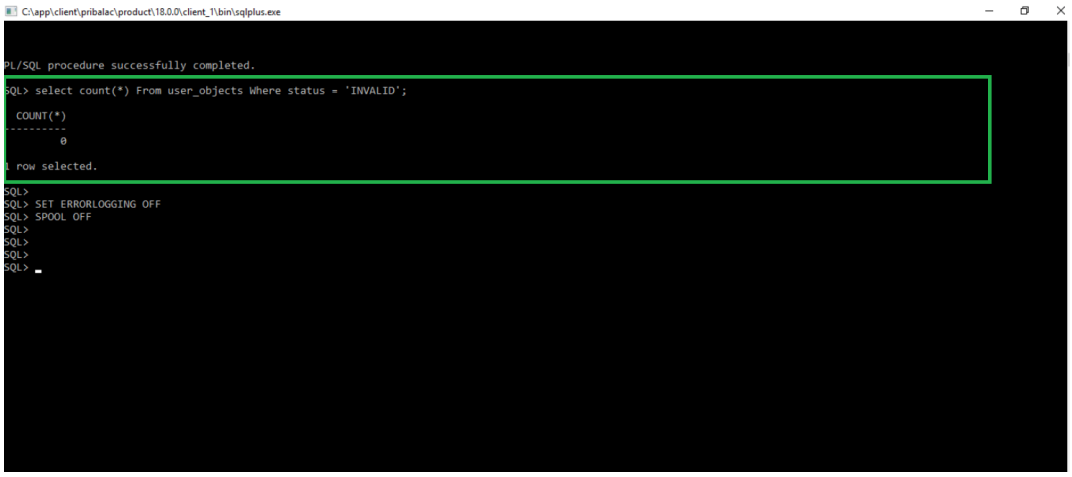
   b. Run `03_Invalids_Recompilation.sql` **for recompiling the invalids during application upgrade**.

   c. No user input is required for this step.

   d. Script will be executed as in the screen shot below and keep the SQL Plus session open for upcoming steps.

   Execution Screenshot:





4. ***End Application upgrade***

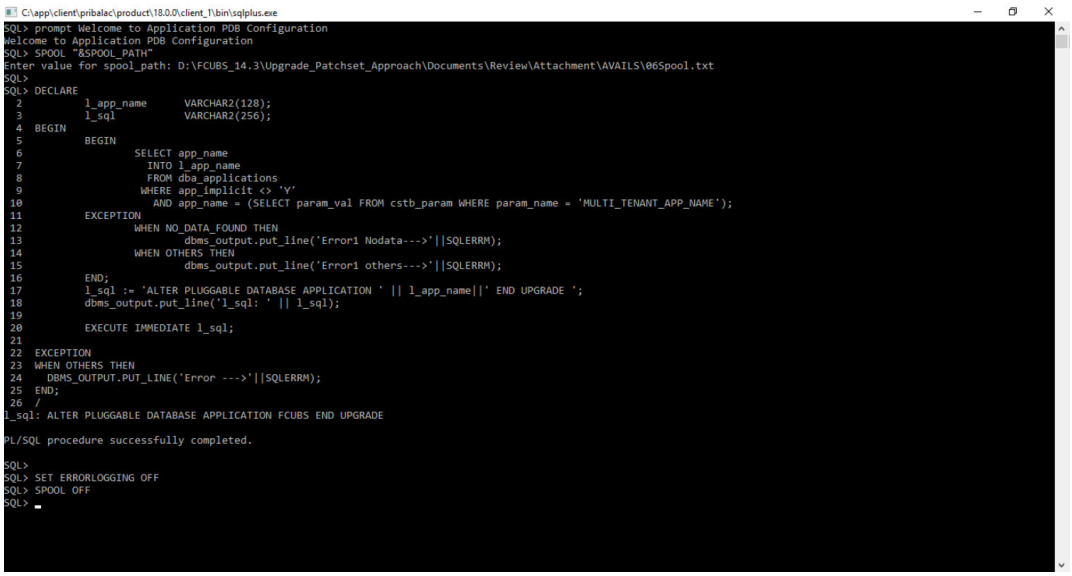
   a. Login into the Approot Schema as Common user.

   b. Run `06_End_Upgrade.sql` **for recompiling the invalids during application upgrade**.

   c. No user input is required for this step.

   d. Script will be executed as in the screen shot below.

Execution Screenshot:



5. ***Start Application upgrade***

   a. Login into the Approot Schema as Common user.

   b. Run `05_Start_Upgrade.sql` **for initiating the application upgrade**.

   c. User input has to be inputted for the below:

Spool Path	<< Any local path>>
Application next version	14.4.0.0.0

   d. Script will be executed similar to step 1 above and keep the SQL Plus session open for upcoming steps.

6. ***Application Root objects conversion for new objects***

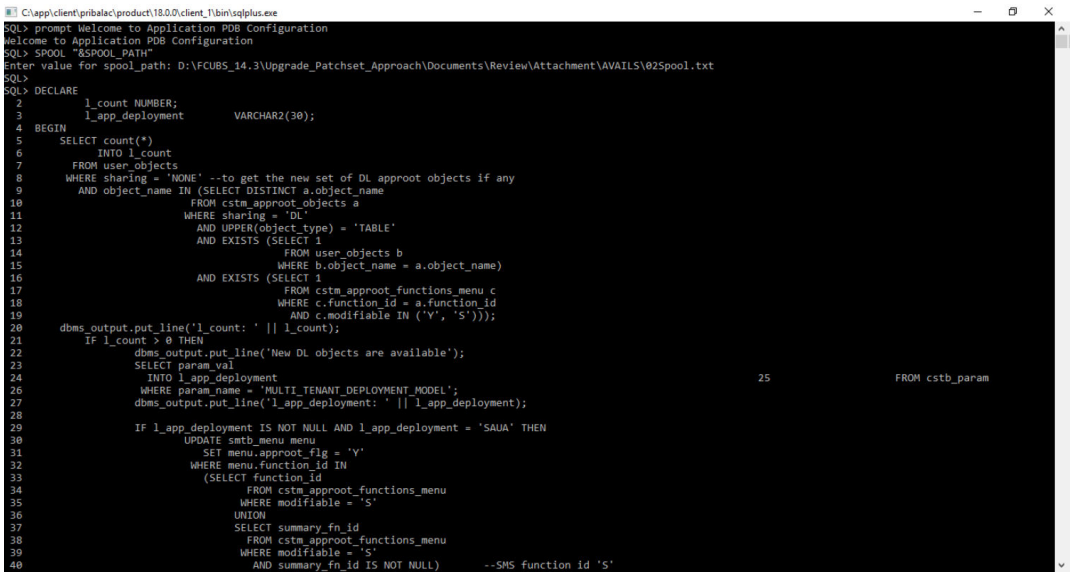
   a. Login into the Approot Schema as Common user.

   b. Run `06_New_Object_Conversion.sql` **for converting new approot objects added during patch-set as DL**

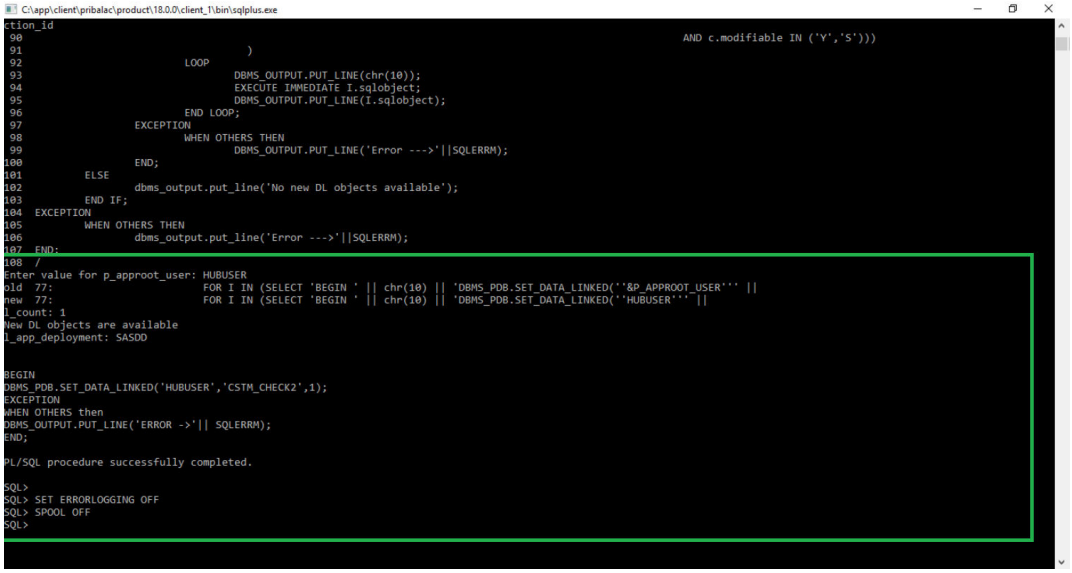
   c. User input has to be inputted for the below:

Spool Path	<< Any local path>>
Application next version	HUBUSER (common user name)

Execution Screenshot:



```
C:\app\client\pribalac\product\18.0.0\client_1\bin\sqlplus.exe
SQL> prompt Welcome to Application PDB Configuration
Welcome to Application PDB Configuration
SQL> SPOOL "&SPOOL_PATH"
Enter value for spool_path: D:\FCUBS_14.3\Upgrade_Patchset_Approach\Documents\Review\Attachment\AVAILS\02Spool.txt
SQL>
SQL> DECLARE
  2         l_count NUMBER;
  3         l_app_deployment        VARCHAR2(30);
  4  BEGIN
  5      SELECT count(*)
  6           INTO l_count
  7        FROM user_objects
  8       WHERE sharing = 'NONE' --to get the new set of DL approot objects if any
  9         AND object_name IN (SELECT DISTINCT a.object_name
 10                             FROM cstm_approot_objects a
 11                            WHERE sharing = 'DL'
 12                              AND UPPER(object_type) = 'TABLE'
 13                              AND EXISTS (SELECT 1
 14                                          FROM user_objects b
 15                                         WHERE b.object_name = a.object_name)
 16                              AND EXISTS (SELECT 1
 17                                          FROM cstm_approot_functions_menu c
 18                                         WHERE c.function_id = a.function_id
 19                                           AND c.modifiable IN ('Y', 'S')));
 20      dbms_output.put_line('l_count: ' || l_count);
 21          IF l_count > 0 THEN
 22                  dbms_output.put_line('New DL objects are available');
 23                  SELECT param_val
 24                    INTO l_app_deployment                                                    25                  FROM cstb_param
 26                   WHERE param_name = 'MULTI_TENANT_DEPLOYMENT_MODEL';
 27                  dbms_output.put_line('l_app_deployment: ' || l_app_deployment);
 28
 29                  IF l_app_deployment IS NOT NULL AND l_app_deployment = 'SAUA' THEN
 30                          UPDATE smtb_menu menu
 31                             SET menu.approot_flg = 'Y'
 32                           WHERE menu.function_id IN
 33                             (SELECT function_id
 34                                     FROM cstm_approot_functions_menu
 35                                    WHERE modifiable = 'S'
 36                                   UNION
 37                                   SELECT summary_fn_id
 38                                     FROM cstm_approot_functions_menu
 39                                    WHERE modifiable = 'S'
 40                                      AND summary_fn_id IS NOT NULL)       --SMS function id 'S'
```



```
C:\app\client\pribalac\product\18.0.0\client_1\bin\sqlplus.exe
ction_id
 90                                                                                  AND c.modifiable IN ('Y','S')))
 91                                  )
 92                          LOOP
 93                                  DBMS_OUTPUT.PUT_LINE(chr(10));
 94                                  EXECUTE IMMEDIATE I.sqlobject;
 95                                  DBMS_OUTPUT.PUT_LINE(I.sqlobject);
 96                          END LOOP;
 97                  EXCEPTION
 98                          WHEN OTHERS THEN
 99                                  DBMS_OUTPUT.PUT_LINE('Error --->'||SQLERRM);
100                  END;
101          ELSE
102                  dbms_output.put_line('No new DL objects available');
103          END IF;
104  EXCEPTION
105          WHEN OTHERS THEN
106                  dbms_output.put_line('Error --->'||SQLERRM);
107  END;
108  /
Enter value for p_approot_user: HUBUSER
old  77:                        FOR I IN (SELECT 'BEGIN ' || chr(10) || 'DBMS_PDB.SET_DATA_LINKED(''&P_APPROOT_USER''' ||
new  77:                        FOR I IN (SELECT 'BEGIN ' || chr(10) || 'DBMS_PDB.SET_DATA_LINKED(''HUBUSER''' ||
l_count: 1
New DL objects are available
l_app_deployment: SASDD


BEGIN
DBMS_PDB.SET_DATA_LINKED('HUBUSER','CSTM_CHECK2',1);
EXCEPTION
WHEN OTHERS then
DBMS_OUTPUT.PUT_LINE('ERROR ->'|| SQLERRM);
END;

PL/SQL procedure successfully completed.

SQL>
SQL> SET ERRORLOGGING OFF
SQL> SPOOL OFF
SQL>
```

7. ***Application Root objects conversion for existing objects***

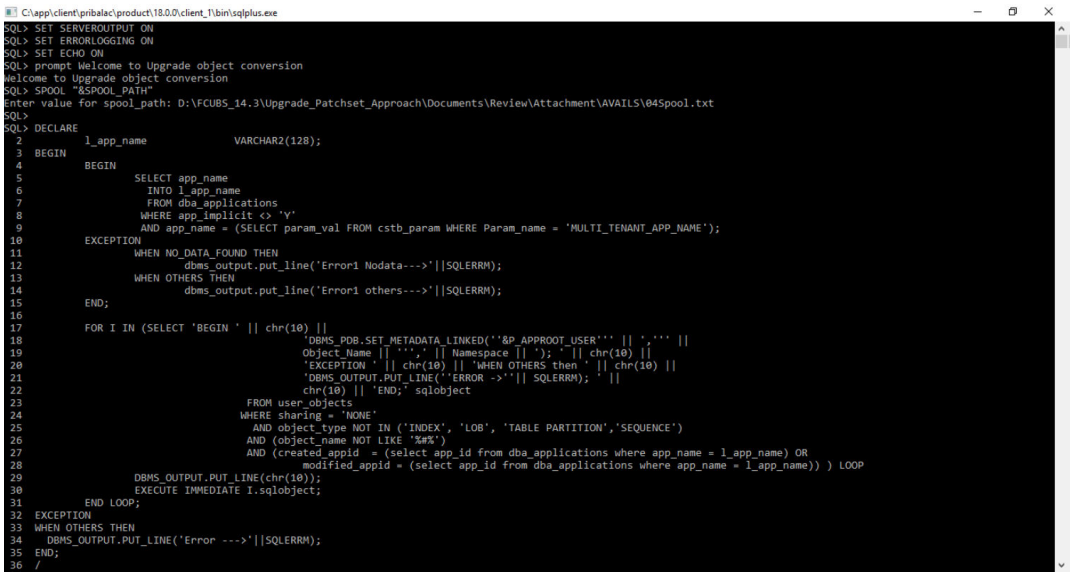
   a. Login into the Approot Schema as Common user.

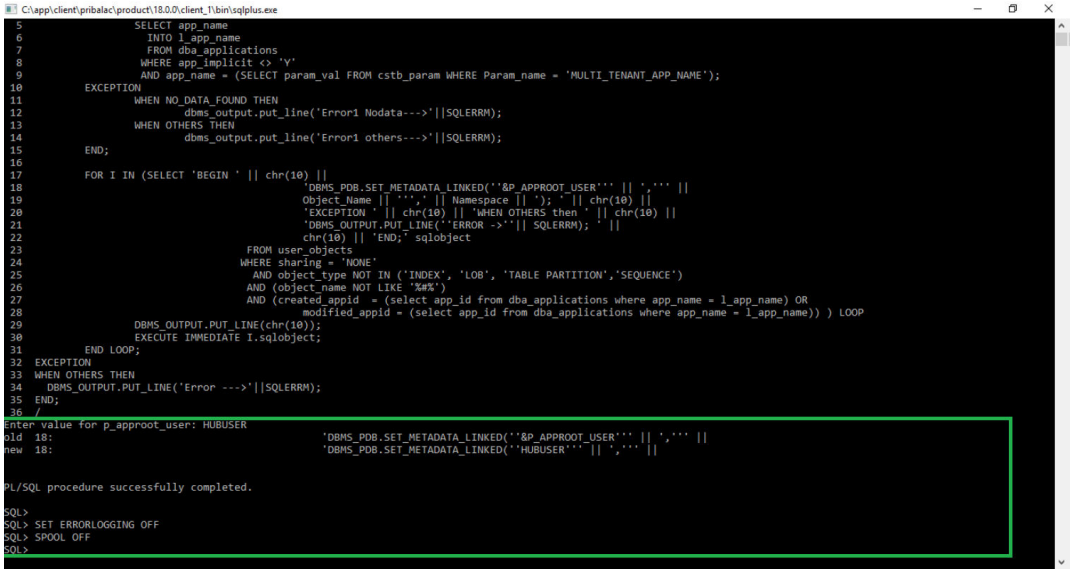
   b. Run `07_Object_Conversion.sql` **for initiating the application upgrade**.

   c. User input has to be inputted for the below:

Spool Path	<< Any local path>>
Application next version	HUBUSER (common user name)

Execution Screenshot:





8. ***Recompilation of invalids***

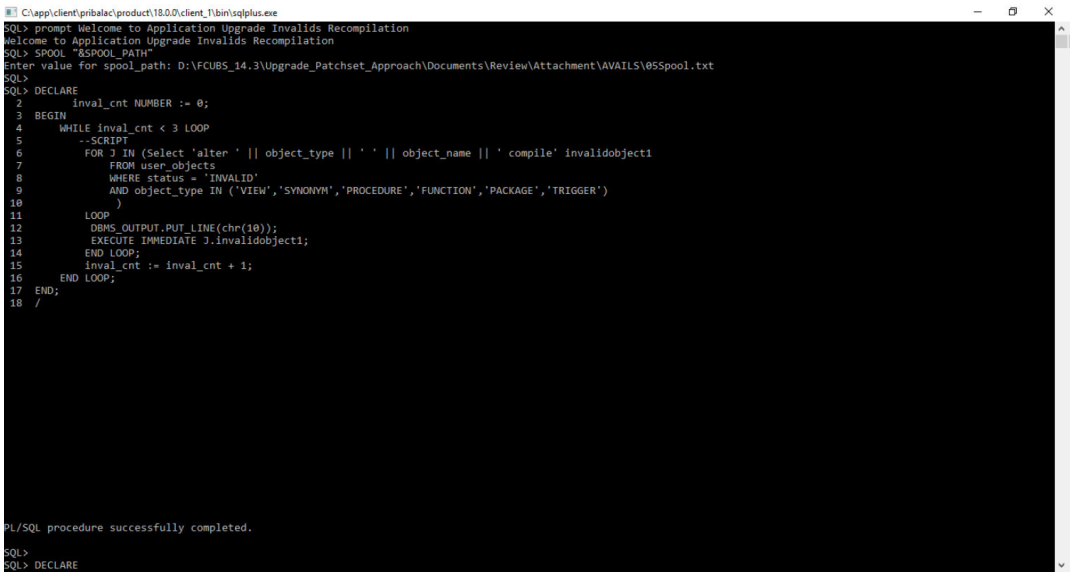
   a. Login into the Approot Schema as Common user.

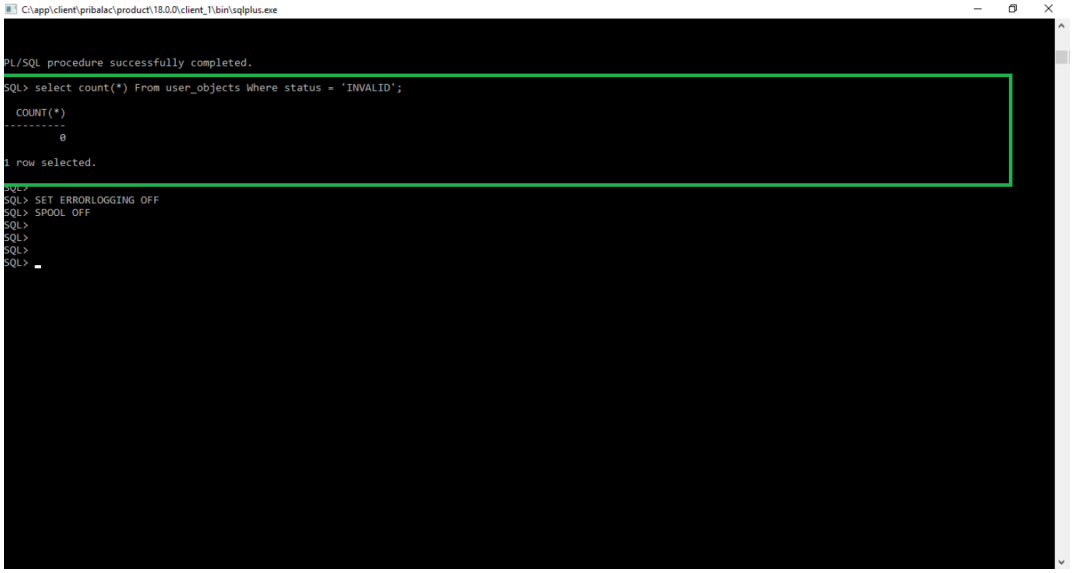
   b. Run `08_Invalids_Recompilation.sql` **for recompiling the invalids during application upgrade**.

   c. No user input is required for this step.

   d. Script will be executed as in the screen shot below and keep the SQL Plus session open for upcoming steps.

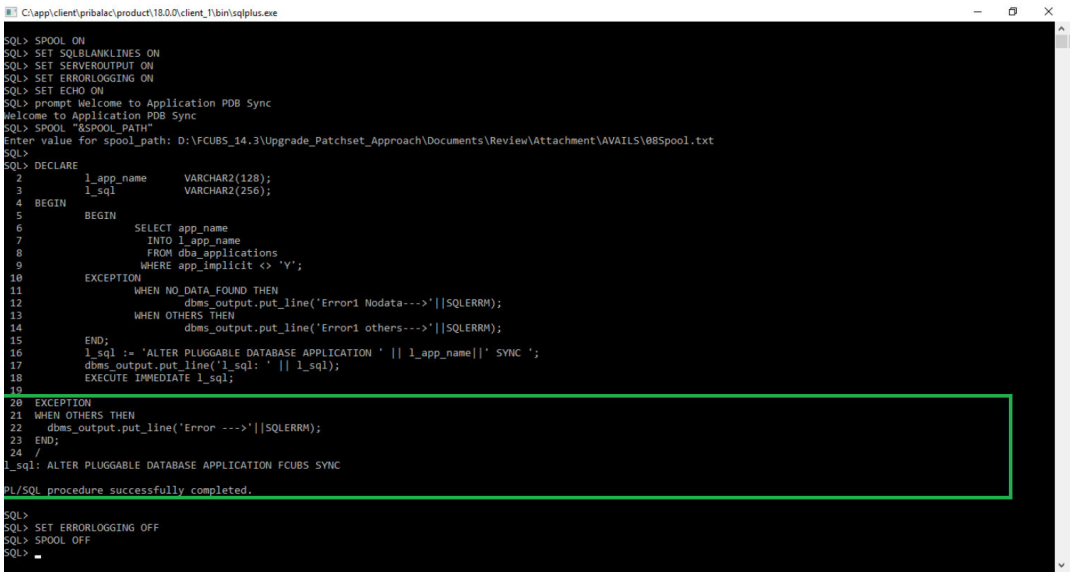
Execution Screenshot:





9. ***End Application upgrade***
   a. Login into the Approot Schema as Common user.
   b. Run `06_End_Upgrade.sql` **for recompiling the invalids during application upgrade**.
   c. No user input is required for this step.
   d. Script will be executed as that of step 4.
10. ***Synchronize application PDBs***
   a. Login into the PDB Schema as Common user. For each PDB, this steps has to be done individually.
   b. Run `07_PDB_Sync.sql` **for synching the application upgrade with PDBs**.
   c. No user input is required for this step.
   d. Script will be executed as in the screen shot below.

Execution Screenshot:



01_Start_Upgrade

This script is used for initiating an application upgrade. This will initiate the application from current version to the next version (patch-set version).

Syntax

```
SET VERIFY ON
SET HEAD ON
SET FEEDBACK 1
SET ARRAY 1
SET LINESIZE 10000
SET PAGESIZE 50000
SET LONG 10000
SET ECHO ON
SET TRIMSPOOL ON
SET COLSEP ';'
SET SERVEROUT OFF
clear screen
SPOOL ON
SET SQLBLANKLINES ON
SET SERVEROUTPUT ON
SET ERRORLOGGING ON
SET ECHO ON
prompt Welcome to Application Upgrade initiation
SPOOL "&SPOOL_PATH"

DECLARE
    l_app_name    VARCHAR2(128);
    l_app_currver VARCHAR2(30);
    l_Sql         VARCHAR2(256);
BEGIN

    BEGIN
        SELECT app_name
          INTO l_app_name
          FROM dba_applications
         WHERE app_implicit <> 'Y'
         AND app_name = (SELECT param_val FROM cstb_param WHERE Param_name =
'MULTI_TENANT_APP_NAME');
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            dbms_output.put_line('Error1 Nodata--->'||SQLERRM);
        WHEN OTHERS THEN
            dbms_output.put_line('Error1 others--->'||SQLERRM);
    END;

    BEGIN
        SELECT MAX(app_version)
          INTO l_app_currver
          FROM dba_app_versions
         WHERE app_name = l_app_name;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
```

```
            dbms_output.put_line('Error2 Nodata--->'||SQLERRM);
        WHEN OTHERS THEN
            dbms_output.put_line('Error2 others--->'||SQLERRM);
    END;

    l_Sql := 'ALTER PLUGGABLE DATABASE APPLICATION ' || l_app_name||' BEGIN
UPGRADE '''|| l_app_currver || ''' TO '''|| '&P_APPLICATION_NEXTVER' ||''''';
    dbms_output.put_line('l_sql: ' || l_Sql);
    EXECUTE IMMEDIATE l_Sql;

    l_Sql := 'ALTER SYSTEM SET DEFAULT_SHARING = NONE';
    dbms_output.put_line('l_sql: ' || l_Sql);
    EXECUTE IMMEDIATE l_Sql;

EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('Error --->'||SQLERRM);
END;
/

SET ERRORLOGGING OFF
SPOOL OFF
```

Recompilation of invalids

Recompilation of objects will be initiated inside the application upgrade for sanity with zero invalids with the below script:

Syntax

```
/*  Script for Shared Application + Shared Data */
SET VERIFY ON
SET HEAD ON
SET FEEDBACK 1
SET ARRAY 1
SET LINESIZE 10000
SET PAGESIZE 50000
SET LONG 10000
SET ECHO ON
SET TRIMSPOOL ON
SET COLSEP ';'
SET SERVEROUT OFF
clear screen
SPOOL ON
SET SQLBLANKLINES ON
SET SERVEROUTPUT ON
SET ERRORLOGGING ON
SET ECHO ON
prompt Welcome to Application Upgrade Invalids Recompilation
SPOOL "&SPOOL_PATH"

DECLARE
    inval_cnt         NUMBER := 0;
    l_object_name     VARCHAR2(240);
BEGIN
    WHILE inval_cnt < 3 LOOP
        --SCRIPT
        FOR J IN (Select 'alter ' || object_type || ' ' || object_name ||'
compile' invalidobject1,
                  object_name
                FROM user_objects
                WHERE status = 'INVALID'
                AND created_appid IS NOT NULL
                AND object_type IN
('VIEW','SYNONYM','PROCEDURE','FUNCTION','PACKAGE','TRIGGER','MATERIALIZED
VIEW'))
        LOOP
            BEGIN
                l_object_name := j.object_name;
                dbms_output.put_line(chr(10));
                EXECUTE IMMEDIATE J.invalidobject1;
            EXCEPTION
                WHEN OTHERS THEN
                    dbms_output.put_line('failed for -->' || l_object_name);
            END;
        END LOOP;
        inval_cnt := inval_cnt + 1;
```

```
    END LOOP;
EXCEPTION
  WHEN OTHERS THEN
        dbms_output.put_line('FAILED FOR -->' || l_object_name);
END;
/
DECLARE
    inval_cnt1         NUMBER := 0;
    l_object_name     VARCHAR2(240);
BEGIN
    WHILE inval_cnt1 < 3 LOOP
        --SCRIPT
        FOR k IN (Select 'alter package '|| object_name||' compile body'
invalidobject2,
                 object_name
                FROM user_objects
                WHERE status = 'INVALID'
                AND created_appid IS NOT NULL
                AND object_type IN ('PACKAGE BODY'))
        LOOP
            BEGIN
                l_object_name := k.object_name;
                dbms_output.put_line(chr(10));
                EXECUTE IMMEDIATE k.invalidobject2;
            EXCEPTION
                WHEN OTHERS THEN
                    dbms_output.put_line('FAILED FOR -->' || l_object_name);
                END;
        END LOOP;
        inval_cnt1 := inval_cnt1 + 1;
    END LOOP;
EXCEPTION
WHEN OTHERS    THEN
    dbms_output.put_line('FAILED FOR -->' || l_object_name);
END;
/
select count(*) From user_objects Where status = 'INVALID';

SET ERRORLOGGING OFF
SPOOL OFF
```

End Application upgrade

This script is run for ending an application upgrade for patch-set.

Pre-requisites:

Step 3 on Application associated pdb creation is completed

Syntax

```
SET VERIFY ON
SET HEAD ON
SET FEEDBACK 1
SET ARRAY 1
SET LINESIZE 10000
SET PAGESIZE 50000
SET LONG 10000
SET ECHO ON
SET TRIMSPOOL ON
SET COLSEP ';'
SET SERVEROUT OFF
clear screen
SPOOL ON
SET SQLBLANKLINES ON
SET SERVEROUTPUT ON
SET ERRORLOGGING ON
SET ECHO ON
prompt Welcome to Application end Upgrade
SPOOL "&SPOOL_PATH"

DECLARE
    l_app_name    VARCHAR2(128);
    l_sql         VARCHAR2(256);
BEGIN
    BEGIN
        SELECT app_name
          INTO l_app_name
          FROM dba_applications
         WHERE app_implicit <> 'Y'
           AND app_name = (SELECT param_val FROM cstb_param WHERE param_name
= 'MULTI_TENANT_APP_NAME');
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            dbms_output.put_line('Error1 Nodata--->'||SQLERRM);
        WHEN OTHERS THEN
            dbms_output.put_line('Error1 others--->'||SQLERRM);
    END;
    l_sql := 'ALTER PLUGGABLE DATABASE APPLICATION ' || l_app_name||' END
UPGRADE ';
    dbms_output.put_line('l_sql: ' || l_sql);

    EXECUTE IMMEDIATE l_sql;

EXCEPTION
```

```
WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE('Error --->'||SQLERRM);
END;
/

SET ERRORLOGGING OFF
SPOOL OFF
```

End Application upgrade

This script runs the invalid script by connecting to the common user in approot outside the upgrade.

Syntax

```
/*  Script for Shared Application + Shared Data */
SET VERIFY ON
SET HEAD ON
SET FEEDBACK 1
SET ARRAY 1
SET LINESIZE 10000
SET PAGESIZE 50000
SET LONG 10000
SET ECHO ON
SET TRIMSPOOL ON
SET COLSEP ';'
SET SERVEROUT OFF
clear screen
SPOOL ON
SET SQLBLANKLINES ON
SET SERVEROUTPUT ON
SET ERRORLOGGING ON
SET ECHO ON
prompt Welcome to Application Upgrade Invalids Recompilation
SPOOL "&SPOOL_PATH"

DECLARE
    inval_cnt         NUMBER := 0;
    l_object_name     VARCHAR2(240);
BEGIN
    WHILE inval_cnt < 3 LOOP
        --SCRIPT
        FOR J IN (Select 'alter ' || object_type || ' ' || object_name ||'
compile' invalidobject1,
                  object_name
                FROM user_objects
                WHERE status = 'INVALID'
                AND created_appid IS NULL
                AND object_type IN
('VIEW','SYNONYM','PROCEDURE','FUNCTION','PACKAGE','TRIGGER','MATERIALIZED
VIEW'))
        LOOP
            BEGIN
                l_object_name := j.object_name;
                dbms_output.put_line(chr(10));
                EXECUTE IMMEDIATE J.invalidobject1;
            EXCEPTION
                WHEN OTHERS THEN
                    dbms_output.put_line('failed for -->' || l_object_name);
            END;
        END LOOP;
        inval_cnt := inval_cnt + 1;
```

```
    END LOOP;
EXCEPTION
  WHEN OTHERS THEN
        dbms_output.put_line('FAILED FOR -->' || l_object_name);
END;
/
DECLARE
    inval_cnt1         NUMBER := 0;
    l_object_name     VARCHAR2(240);
BEGIN
    WHILE inval_cnt1 < 3 LOOP
        --SCRIPT
        FOR k IN (Select 'alter package '|| object_name||' compile body'
invalidobject2,
                 object_name
                FROM user_objects
                WHERE status = 'INVALID'
                AND created_appid IS NULL
                AND object_type IN ('PACKAGE BODY'))
        LOOP
            BEGIN
                l_object_name := k.object_name;
                dbms_output.put_line(chr(10));
                EXECUTE IMMEDIATE k.invalidobject2;
            EXCEPTION
                WHEN OTHERS THEN
                    dbms_output.put_line('FAILED FOR -->' || l_object_name);
                END;
        END LOOP;
        inval_cnt1 := inval_cnt1 + 1;
    END LOOP;
EXCEPTION
WHEN OTHERS    THEN
    dbms_output.put_line('FAILED FOR -->' || l_object_name);
END;
/
select count(*) From user_objects Where status = 'INVALID';

SET ERRORLOGGING OFF
SPOOL OFF
```

Start Application upgrade

This script is run to initiate another application upgrade for object conversion.

Syntax

```
SET VERIFY ON
SET HEAD ON
SET FEEDBACK 1
SET ARRAY 1
SET LINESIZE 10000
SET PAGESIZE 50000
SET LONG 10000
SET ECHO ON
SET TRIMSPOOL ON
SET COLSEP ';'
SET SERVEROUT OFF
clear screen
SPOOL ON
SET SQLBLANKLINES ON
SET SERVEROUTPUT ON
SET ERRORLOGGING ON
SET ECHO ON
prompt Welcome to Application Upgrade initiation
SPOOL "&SPOOL_PATH"

DECLARE
    l_app_name    VARCHAR2(128);
    l_app_currver VARCHAR2(30);
    l_Sql         VARCHAR2(256);
BEGIN

    BEGIN
        SELECT app_name
          INTO l_app_name
          FROM dba_applications
         WHERE app_implicit <> 'Y'
         AND app_name = (SELECT param_val FROM cstb_param WHERE Param_name =
'MULTI_TENANT_APP_NAME');
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            dbms_output.put_line('Error1 Nodata--->'||SQLERRM);
        WHEN OTHERS THEN
            dbms_output.put_line('Error1 others--->'||SQLERRM);
    END;

    BEGIN
        SELECT MAX(app_version)
          INTO l_app_currver
          FROM dba_app_versions
         WHERE app_name = l_app_name;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            dbms_output.put_line('Error2 Nodata--->'||SQLERRM);
```

```
        WHEN OTHERS THEN
            dbms_output.put_line('Error2 others--->'||SQLERRM);
    END;

    l_Sql := 'ALTER PLUGGABLE DATABASE APPLICATION ' || l_app_name||' BEGIN
UPGRADE '''|| l_app_currver || ''' TO '''|| '&P_APPLICATION_NEXTVER' ||''''';
    dbms_output.put_line('l_sql: ' || l_Sql);
    EXECUTE IMMEDIATE l_Sql;

    l_Sql := 'ALTER SYSTEM SET DEFAULT_SHARING = NONE';
    dbms_output.put_line('l_sql: ' || l_Sql);
    EXECUTE IMMEDIATE l_Sql;

EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('Error --->'||SQLERRM);
END;
/

SET ERRORLOGGING OFF
SPOOL OFF
```

Application Root objects conversion for new objects

This is script is used in converting the new DL objects during patch-set based on the deployment model of the application during installation.

Syntax

```
/*    Script    for    Shared    Object    Conversion    for    patch-set
*/
SET VERIFY ON
SET HEAD ON
SET FEEDBACK 1
SET ARRAY 1
SET LINESIZE 10000
SET PAGESIZE 50000
SET LONG 10000
SET ECHO ON
SET TRIMSPOOL ON
SET COLSEP ';'
SET SERVEROUT OFF
clear screen
SPOOL ON
SET SQLBLANKLINES ON
SET SERVEROUTPUT ON
SET ERRORLOGGING ON
SET ECHO ON
prompt Welcome to Upgrade New object Conversion
SPOOL    "&SPOOL_PATH"

DECLARE
    l_count          NUMBER;
    l_app_deployment VARCHAR2(30);
BEGIN
    SELECT count(*)
      INTO l_count
      FROM user_objects
     WHERE sharing = 'NONE' --to get the new set of DL approot objects if any
       AND object_name IN
           (SELECT DISTINCT a.object_name
              FROM cstm_approot_objects a
             WHERE sharing = 'DL'
               AND UPPER(object_type) = 'TABLE'
               AND EXISTS (SELECT 1
                     FROM user_objects b
                    WHERE b.object_name = a.object_name)
               AND EXISTS (SELECT 1
                     FROM cstm_approot_functions_menu c
                    WHERE c.function_id = a.function_id
                      AND c.modifiable IN ('Y', 'S')));
    dbms_output.put_line('l_count:    '    ||    l_count);

    IF l_count > 0 THEN
```

```
        dbms_output.put_line('New DL objects are available');
        SELECT param_val
          INTO l_app_deployment
          FROM cstb_param
         WHERE param_name = 'MULTI_TENANT_DEPLOYMENT_MODEL';
        dbms_output.put_line('l_app_deployment: '||l_app_deployment);

        IF l_app_deployment IS NOT NULL AND l_app_deployment = 'SAUA' THEN
            UPDATE smtb_menu menu
               SET menu.approot_flg = 'Y'
             WHERE menu.function_id IN
                   (SELECT function_id
                      FROM cstm_approot_functions_menu
                     WHERE modifiable = 'S'
                     UNION
                     SELECT summary_fn_id
                       FROM cstm_approot_functions_menu
                      WHERE modifiable = 'S'
                        AND summary_fn_id IS NOT NULL) --SMS function id 'S'
               AND menu.approot_flg <> 'Y'; --excluding the already modified
approot function ids in menu.
        ELSIF l_app_deployment IS NOT NULL AND l_app_deployment = 'SASDD' THEN
            UPDATE smtb_menu menu
               SET menu.approot_flg = 'Y'
             WHERE menu.function_id IN
                   (SELECT function_id
                      FROM cstm_approot_functions_menu
                     UNION
                     SELECT summary_fn_id
                       FROM cstm_approot_functions_menu
                      WHERE summary_fn_id IS NOT NULL)
               AND menu.approot_flg <> 'Y'; --excluding the already modified
approot function ids in menu.
        ELSIF l_app_deployment IS NOT NULL AND l_app_deployment = 'SASDC' THEN
        /*Assumption new table cstm_approot_menu_custom_movedtopdb will be
available
            and is populated with the function ids which are moved to PDB as
part of custom deployment
            It has 2 columns FUNCTION_ID and SUMMARY_FN_ID*/

            UPDATE smtb_menu menu
               SET menu.approot_flg = 'Y'
             WHERE menu.function_id IN
                   (SELECT function_id
                      FROM cstm_approot_functions_menu
                     UNION
                     SELECT summary_fn_id
                       FROM cstm_approot_functions_menu
                      WHERE summary_fn_id IS NOT NULL)
               AND menu.function_id NOT IN --excluding the function ids moved
to PDB already.
                   (SELECT function_id
                      FROM cstm_approot_menu_movedtopdb
                     UNION
                     SELECT summary_fn_id
                       FROM cstm_approot_menu_movedtopdb
```

```
                    WHERE summary_fn_id IS NOT NULL)
                AND menu.approot_flg <> 'Y'; --excluding the already modified
approot function ids in menu.
        END IF;

        BEGIN
            FOR I IN (SELECT 'BEGIN ' || chr(10) ||
                             'DBMS_PDB.SET_DATA_LINKED(''&P_APPROOT_USER''' ||
',''' ||
                             Object_Name || ''',' || Namespace || '); ' ||
chr(10) ||
                             'EXCEPTION ' || chr(10) ||
                             'WHEN OTHERS THEN ' || chr(10) ||
                             'DBMS_OUTPUT.PUT_LINE(''ERROR ->''|| SQLERRM); ' ||
                             chr(10) || 'END;' sqlobject
                      FROM user_objects
                     WHERE sharing = 'NONE' --to get the new set of DL
approot objects if any
                       AND object_name IN
                             (SELECT DISTINCT a.object_name
                                FROM cstm_approot_objects a
                               WHERE sharing = 'DL'
                                 AND UPPER(object_type) = 'TABLE'
                                 AND EXISTS (SELECT 1
                                               FROM user_objects b
                                              WHERE b.object_name = a.object_name)
                                 AND EXISTS
                                (SELECT 1
                                                FROM cstm_approot_functions_menu c
                                               WHERE c.function_id = a.function_id
                                                 AND c.modifiable IN ('Y', 'S')))) LOOP
                DBMS_OUTPUT.PUT_LINE(chr(10));
                EXECUTE IMMEDIATE I.sqlobject;
                DBMS_OUTPUT.PUT_LINE(I.sqlobject);
            END LOOP;
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('Error --->' || SQLERRM);
        END;
    ELSE
        dbms_output.put_line('No new DL objects available');
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('Error --->'||SQLERRM);
END;
/

SET ERRORLOGGING OFF
SPOOL OFF
```

Object_Conversion

Purpose

This script is used for converting the modified MDL objects when there is a re-creation [objects with Create or Replace command during creation] happens during patch-set

Syntax

```
SET VERIFY ON
SET HEAD ON
SET FEEDBACK 1
SET ARRAY 1
SET LINESIZE 10000
SET PAGESIZE 50000
SET LONG 10000
SET ECHO ON
SET TRIMSPOOL ON
SET COLSEP ';'
SET SERVEROUT OFF
clear screen
SPOOL ON
SET SQLBLANKLINES ON
SET SERVEROUTPUT ON
SET ERRORLOGGING ON
SET ECHO ON
prompt Welcome to Upgrade object conversion
SPOOL "&SPOOL_PATH"
BEGIN
    FOR I IN (SELECT 'BEGIN ' || chr(10) ||
                        'DBMS_PDB.SET_METADATA_LINKED(''&P_APPROOT_USER''' ||
',''' ||
                        Object_Name || ''',' || Namespace || '); ' || chr(10)
||
                        'EXCEPTION ' || chr(10) || 'WHEN OTHERS then ' ||
chr(10) ||
                        'DBMS_OUTPUT.PUT_LINE(''ERROR ->''|| SQLERRM); ' ||
                        chr(10) || 'END;' sqlobject
                   FROM user_objects
                  WHERE sharing = 'NONE'
                    AND object_type NOT IN ('INDEX', 'LOB', 'TABLE
PARTITION','SEQUENCE','JOB','MATERIALIZED VIEW','MATERIALIZED VIEW LOG')
                    AND application = 'Y'
                    AND (object_name,object_type) NOT IN (SELECT
object_name,object_type
                                                          FROM
cstm_approot_objects
                                                          WHERE  function_id
= 'DYNAMIC'
                                                          AND    sharing
= 'NONE'
                                                         )
             ) LOOP
        dbms_output.put_line(chr(10));
```

```
        EXECUTE IMMEDIATE I.sqlobject;
        dbms_output.put_line(I.sqlobject);
    END LOOP;
EXCEPTION
WHEN OTHERS THEN
  dbms_output.put_line('Error --->'||SQLERRM);
END;
/
SET ERRORLOGGING OFF
SPOOL OFF
```

Invalids_Recompilation_Inside_Upgrade

Purpose

This script is used to initiate the recompilation of objects inside the application upgrade for sanity.

Syntax

```
/*  Script for Shared Application + Shared Data */
SET VERIFY ON
SET HEAD ON
SET FEEDBACK 1
SET ARRAY 1
SET LINESIZE 10000
SET PAGESIZE 50000
SET LONG 10000
SET ECHO ON
SET TRIMSPOOL ON
SET COLSEP ';'
SET SERVEROUT OFF
clear screen
SPOOL ON
SET SQLBLANKLINES ON
SET SERVEROUTPUT ON
SET ERRORLOGGING ON
SET ECHO ON
prompt Welcome to Application Upgrade Invalids Recompilation
SPOOL "&SPOOL_PATH"

DECLARE
    inval_cnt         NUMBER := 0;
    l_object_name     VARCHAR2(240);
BEGIN
    WHILE inval_cnt < 3 LOOP
        --SCRIPT
        FOR J IN (Select 'alter ' || object_type || ' ' || object_name ||'
compile' invalidobject1,
                  object_name
                FROM user_objects
                WHERE status = 'INVALID'
                AND created_appid IS NOT NULL
                AND object_type IN
('VIEW','SYNONYM','PROCEDURE','FUNCTION','PACKAGE','TRIGGER','MATERIALIZED
VIEW'))
        LOOP
            BEGIN
                l_object_name := j.object_name;
                dbms_output.put_line(chr(10));
                EXECUTE IMMEDIATE J.invalidobject1;
            EXCEPTION
                WHEN OTHERS THEN
                    dbms_output.put_line('failed for -->' || l_object_name);
            END;
```

```
        END LOOP;
        inval_cnt := inval_cnt + 1;
    END LOOP;
EXCEPTION
  WHEN OTHERS THEN
        dbms_output.put_line('FAILED FOR -->' || l_object_name);
END;
/
DECLARE
    inval_cnt1         NUMBER := 0;
    l_object_name     VARCHAR2(240);
BEGIN
    WHILE inval_cnt1 < 3 LOOP
        --SCRIPT
        FOR k IN (Select 'alter package '|| object_name||' compile body'
invalidobject2,
                 object_name
                FROM user_objects
                WHERE status = 'INVALID'
                AND created_appid IS NOT NULL
                AND object_type IN ('PACKAGE BODY'))
        LOOP
            BEGIN
                l_object_name := k.object_name;
                dbms_output.put_line(chr(10));
                EXECUTE IMMEDIATE k.invalidobject2;
            EXCEPTION
                WHEN OTHERS THEN
                    dbms_output.put_line('FAILED FOR -->' || l_object_name);
                END;
        END LOOP;
        inval_cnt1 := inval_cnt1 + 1;
    END LOOP;
EXCEPTION
WHEN OTHERS    THEN
    dbms_output.put_line('FAILED FOR -->' || l_object_name);
END;
/
select count(*) From user_objects Where status = 'INVALID';

SET ERRORLOGGING OFF
SPOOL OFF
```

End_Upgrade

Purpose

This script is run for ending an application upgrade for patch-set.

Syntax

```
/* Pre-requisites: Step 3 on Application associated pdb creation is completed
*/
SET VERIFY ON
SET HEAD ON
SET FEEDBACK 1
SET ARRAY 1
SET LINESIZE 10000
SET PAGESIZE 50000
SET LONG 10000
SET ECHO ON
SET TRIMSPOOL ON
SET COLSEP ';'
SET SERVEROUT OFF
clear screen
SPOOL ON
SET SQLBLANKLINES ON
SET SERVEROUTPUT ON
SET ERRORLOGGING ON
SET ECHO ON
prompt Welcome to Application end Upgrade
SPOOL "&SPOOL_PATH"

DECLARE
    l_app_name VARCHAR2(128);
    l_sql VARCHAR2(256);
BEGIN
    BEGIN
        SELECT app_name
          INTO l_app_name
          FROM dba_applications
         WHERE app_implicit <> 'Y'
           AND app_name = (SELECT param_val FROM cstb_param WHERE param_name
= 'MULTI_TENANT_APP_NAME');
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            dbms_output.put_line('Error1 Nodata--->'||SQLERRM);
        WHEN OTHERS THEN
            dbms_output.put_line('Error1 others--->'||SQLERRM);
    END;
    l_sql := 'ALTER PLUGGABLE DATABASE APPLICATION ' || l_app_name||' END
UPGRADE ';
    dbms_output.put_line('l_sql: ' || l_sql);

    EXECUTE IMMEDIATE l_sql;

EXCEPTION
```

```
WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE('Error --->'||SQLERRM);
END;
/

SET ERRORLOGGING OFF
SPOOL OFF
```

Invalids_Recompilation_Outside_Upgrade.sql

Purpose

This script is run for invalid script by connecting to the common user in approot outside the upgrade.

Syntax

```
/*  Script for Shared Application + Shared Data */
SET VERIFY ON
SET HEAD ON
SET FEEDBACK 1
SET ARRAY 1
SET LINESIZE 10000
SET PAGESIZE 50000
SET LONG 10000
SET ECHO ON
SET TRIMSPOOL ON
SET COLSEP ';'
SET SERVEROUT OFF
clear screen
SPOOL ON
SET SQLBLANKLINES ON
SET SERVEROUTPUT ON
SET ERRORLOGGING ON
SET ECHO ON
prompt Welcome to Application Upgrade Invalids Recompilation
SPOOL "&SPOOL_PATH"

DECLARE
    inval_cnt         NUMBER := 0;
    l_object_name     VARCHAR2(240);
BEGIN
    WHILE inval_cnt < 3 LOOP
        --SCRIPT
        FOR J IN (Select 'alter ' || object_type || ' ' || object_name ||'
compile' invalidobject1,
                  object_name
                FROM user_objects
                WHERE status = 'INVALID'
                AND created_appid IS NULL
                AND object_type IN
('VIEW','SYNONYM','PROCEDURE','FUNCTION','PACKAGE','TRIGGER','MATERIALIZED
VIEW'))
        LOOP
            BEGIN
                l_object_name := j.object_name;
                dbms_output.put_line(chr(10));
                EXECUTE IMMEDIATE J.invalidobject1;
            EXCEPTION
                WHEN OTHERS THEN
                    dbms_output.put_line('failed for -->' || l_object_name);
            END;
```

```
        END LOOP;
        inval_cnt := inval_cnt + 1;
    END LOOP;
EXCEPTION
  WHEN OTHERS THEN
        dbms_output.put_line('FAILED FOR -->' || l_object_name);
END;
/
DECLARE
    inval_cnt1         NUMBER := 0;
    l_object_name     VARCHAR2(240);
BEGIN
    WHILE inval_cnt1 < 3 LOOP
        --SCRIPT
        FOR k IN (Select 'alter package '|| object_name||' compile body'
invalidobject2,
                 object_name
                FROM user_objects
                WHERE status = 'INVALID'
                AND created_appid IS NULL
                AND object_type IN ('PACKAGE BODY'))
        LOOP
            BEGIN
                l_object_name := k.object_name;
                dbms_output.put_line(chr(10));
                EXECUTE IMMEDIATE k.invalidobject2;
            EXCEPTION
                WHEN OTHERS THEN
                    dbms_output.put_line('FAILED FOR -->' || l_object_name);
                END;
        END LOOP;
        inval_cnt1 := inval_cnt1 + 1;
    END LOOP;
EXCEPTION
WHEN OTHERS    THEN
    dbms_output.put_line('FAILED FOR -->' || l_object_name);
END;
/
select count(*) From user_objects Where status = 'INVALID';

SET ERRORLOGGING OFF
SPOOL OFF
```

PDB_Sync

Purpose

This script is run to synchronize the PDBs with the latest application changes in the application root.

Syntax

```
SET VERIFY ON
SET HEAD ON
SET FEEDBACK 1
SET ARRAY 1
SET LINESIZE 10000
SET PAGESIZE 50000
SET LONG 10000
SET ECHO ON
SET TRIMSPOOL ON
SET COLSEP ';'
SET SERVEROUT OFF
clear screen
SPOOL ON
SET SQLBLANKLINES ON
SET SERVEROUTPUT ON
SET ERRORLOGGING ON
SET ECHO ON
prompt Welcome to Application PDB Sync
SPOOL "&SPOOL_PATH"

DECLARE
    l_app_name VARCHAR2(128);
    l_sql VARCHAR2(256);
BEGIN
    BEGIN
        SELECT app_name
          INTO l_app_name
          FROM dba_applications
         WHERE app_implicit <> 'Y';
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            dbms_output.put_line('Error1 Nodata--->'||SQLERRM);
        WHEN OTHERS THEN
            dbms_output.put_line('Error1 others--->'||SQLERRM);
    END;
    l_sql := 'ALTER PLUGGABLE DATABASE APPLICATION ' || l_app_name||' SYNC ';
    dbms_output.put_line('l_sql: ' || l_sql);
    EXECUTE IMMEDIATE l_sql;

EXCEPTION
WHEN OTHERS THEN
  dbms_output.put_line('Error --->'||SQLERRM);
END;
/
```

```
SET ERRORLOGGING OFF
SPOOL OFF
```

Glossary

Index

A

P