Oracle Banking Trade Finance

Multi-Tenant Deployment

Release 14.8.1.0.0
G46015-01
October 2025

ORACLE

Oracle Banking Trade Finance Multi-Tenant Deployment, Release 14.8.1.0.0

G46015-01

Contents

Preface

Purpose

This manual is designed to help you quickly get acquainted with Multi-Tenant Deployment.

Audience

This guide is intended for anyone responsible for installing Oracle Banking Application.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Critical Patches

Oracle advises customers to get all their security vulnerability information from the Oracle Critical Patch Update Advisory, which is available at Critical Patches, Security Alerts and Bulletins. All critical patches should be applied in a timely manner to ensure effective security, as strongly recommended by Oracle Software Security Assurance.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners,

we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Related Resources

For more information, see these Oracle Banking Trade Finance resources:

- Oracle Banking Trade Finance Release Notes
- Oracle Banking Trade Finance Install & Upgrade

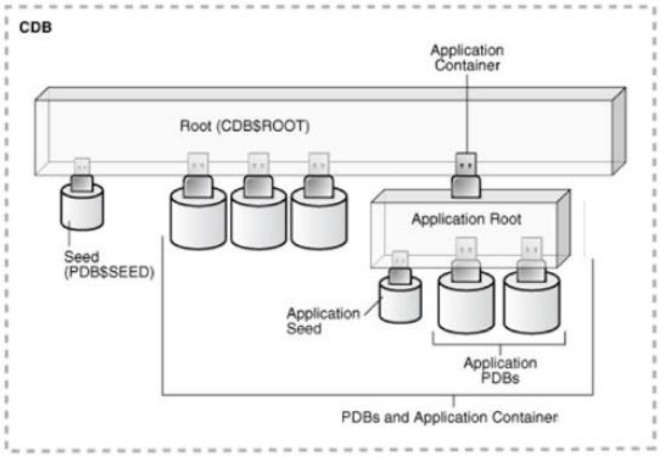
Screenshot Disclaimer

Personal information used in the interface or documents is dummy and does not exist in the real world. It is only for reference purposes.

1

Oracle Multi-Tenant Architecture

Figure 1-1 Multi-Tenant Architecture



- Overview of the Multitenant Architecture
- Application Maintenance
  Application maintenance refers to installing, uninstalling, upgrading, or patching an application.

1.1 Overview of the Multitenant Architecture

- Container Database
  The CDB is a collection of schemas, schema objects, and non-schema objects to which all PDBs belong.
- Application Root
- Seed PDB
- Application PDB

1.1.1 Container Database

The CDB is a collection of schemas, schema objects, and non-schema objects to which all PDBs belong.

Every CDB has one and only one root container named CDB$ROOT. The root stores the system metadata required to manage PDBs. All PDBs belong to the root. The system container is the CDB root and all PDBs that belong to this root.

The CDB root does not store user data. Oracle recommends that you do not add common objects to the root or modify Oracle-supplied schemas in the root. However, you can create common users and roles for database administration. A common user with the necessary privileges can switch between containers.

Section Title

(Optional) Enter conceptual text here.

Example 1-1 Example Title

(Optional) Enter an example here.

1.1.2 Application Root

Consider an application root as an application-specific root container. It serves as a repository for a master definition of an application back end, including common data and metadata. To create an application root, connect to the CDB root and specify the AS APPLICATION CONTAINER clause in a CREATE PLUGGABLE DATABASE statement.

1.1.3 Seed PDB

Unlike a standard PDB, a seed PDB is not intended to support an application. Rather, the seed is a template for the creation of PDBs that support applications. To accelerate creation of application PDBs within an application container, you can create an application seed. An application container contains either zero or one application seed.

1.1.4 Application PDB

An application PDB belongs to exactly one application container. Unlike PDBs plugged in to the CDB root, application PDBs can share a master application definition within an application container. For example, a user_details table in an application root might be a data-linked common object, which means it contains data accessible by all application PDBs plugged in to this root. PDBs that do not reside within the application container cannot access its application common objects.

1.2 Application Maintenance

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

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.

- 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.
- 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.

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

- Name of the application
- Application version number

1.2.2 Application Upgrade

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

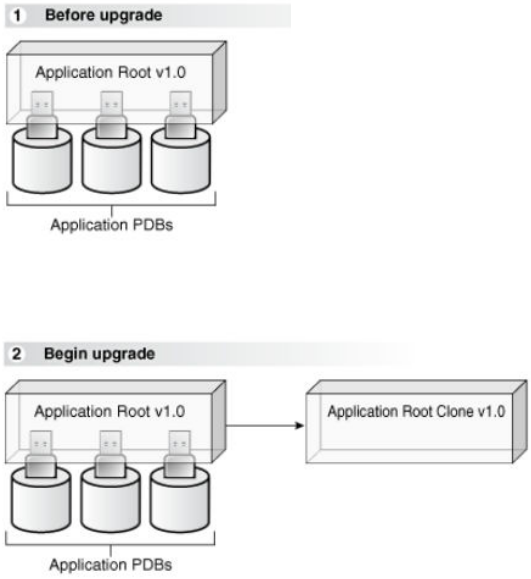
Typically, an upgrade changes the physical architecture of the application. For example, an upgrade might add new user accounts, tables, and packages, or alter the definitions of existing objects.

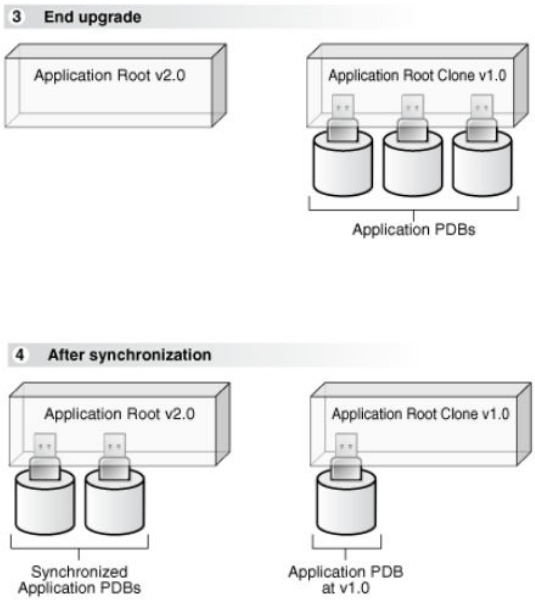
To upgrade the application, you must specify the following in the ALTER PLUGGABLE DATABASE APPLICATION statement:

- 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.





2

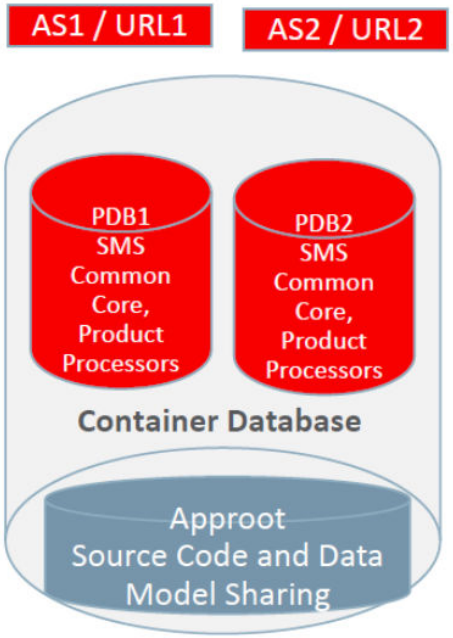
Proposed Deployment Model

- Shared Application
  In this model application would be deployed in an application container in 19C, Multiple front-end applications with URL is created per PDB.
- Shared Application and User Authentication
  In this model application would be deployed in an application container in 19C, Single front end application and an URL.
- Shared Application with Shared Data - Default
  This would be using Application Container in 19C, Single front end application and an URL. Sharing of Entities from Approot to individual PDBs.
- Shared Application with Shared Data - Custom
  This would be using Application Container in 19C, Single front end application and an URL. Sharing of Entities from Approot to individual PDBs.

2.1 Shared Application

In this model application would be deployed in an application container in 19C, Multiple front-end applications with URL is created per PDB.

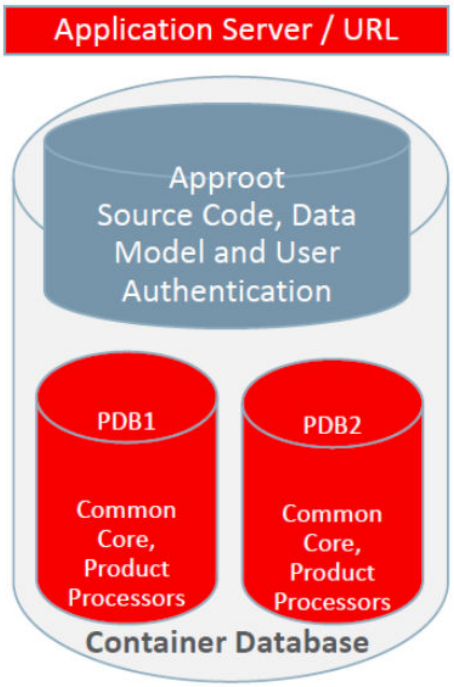
- Application would be deployed in an Application Container.
- Source code at Approot level shared with PDBs.
- Data Model at Approot level shared with PDBs.
- No sharing of data
- Multiple frontend application with URL per PDB (with common EAR file).



2.2 Shared Application and User Authentication

In this model application would be deployed in an application container in 19C, Single front end application and an URL.

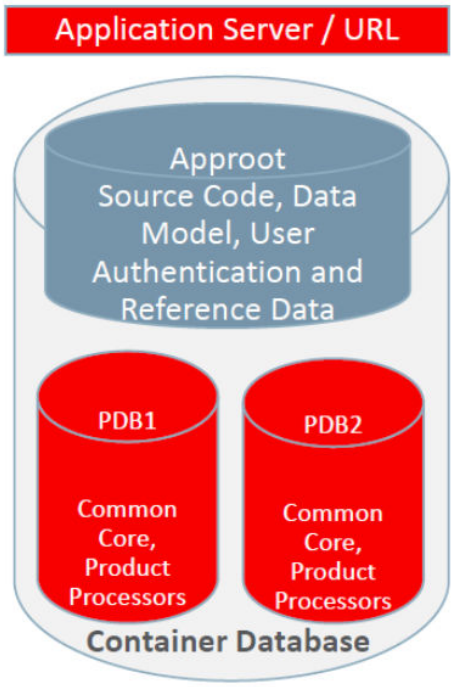
- Application would be deployed in an Application Container.
- Source code at Approot level shared with PDBs.
- Data Model at Approot level shared with PDBs.
- Sharing of data related to User Authentication.
- Single Frontend Application and Single URL.



2.3 Shared Application with Shared Data - Default

This would be using Application Container in 19C, Single front end application and an URL. Sharing of Entities from Approot to individual PDBs.

- Application would be deployed in an Application Container.
- Source code at Approot level shared with PDBs.
- Data Model at Approot level shared with PDBs.
- Single Frontend Application and Single URL.
- Sharing of Entities/data like.
  - User Authentication, SMS Roles
  - Core Entities like Country, Currency, MIS Classes, UDFs
  - Chart of Account, Product, Account Class



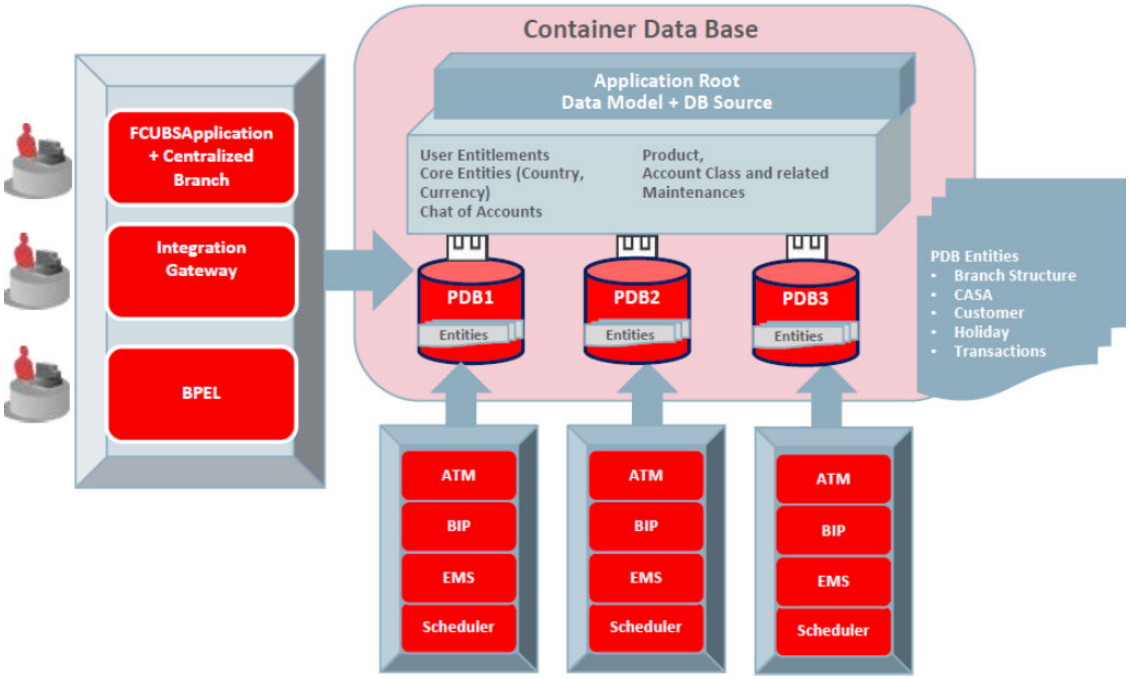
2.4 Shared Application with Shared Data - Custom

This would be using Application Container in 19C, Single front end application and an URL. Sharing of Entities from Approot to individual PDBs.

- Application would be deployed in an Application Container.
- Source code at Approot level shared with PDBs.
- Data Model at Approot level shared with PDBs.
- Single Frontend Application and Single URL.
- Sharing of Entities/data like.
  - User Authentication, SMS Roles
  - Core Entities like Country, Currency, MIS Classes, UDFs
  - Chart of Account, Product, Account Class
- User can opt-out the entities which are not required to be the candidates of approot and move to PDB.

Sample of components deployed in Shared Application and Shared Data model is given below:

Figure 2-1 Component Deployment Architecture



Application and Gateway will be common and single URL will be available for the application. ATM, BIP, EMS, Scheduler has to be configured separately for each PDBs.

3 Deployment and Installation Steps

As a pre-requisite, DB server has to be created with 18c database installed along with CDB setup.

Multi entity application root/PDB based application setup can be done by following below steps in sequential order, and detail of each steps explained as separate section subsequently.

1. Application Template PDB configuration
   - Application Template PDB is a normal PDB created under CDB to install the required DB objects for a product processor. This PDB will have a common schema and is used as a template for creating Application root through cloning.
2. Application root and Application Seed configuration
   a. Application root
      i. Application root is an application-specific root container and repositories for an application back end DB objects.
      ii. Application root will be created through cloning from Application Template PDB.
   b. Application Seed
      i. Application seed is created to accelerate the creation of application PDBs within an application container.
      ii. Application seed will be created from Application root through cloning and used as template to create one or more Application PDBs.
3. Application Installation
   a. Application installation has to be done in the approot as version 1.0 with being user made explicit.
4. Application Root objects conversion
   a. All the DB objects loaded in Application root will be converted as DATA LINK or METADATA LINK.
5. Application Seed Sync with the Application Root
   a. Any changes deployed in Application Root will be available at Application PDB, if Application PDB sync with Application Root.

> **Note**
>
> Application root should be synced with application seed always. It will help during the creation of new application PDBs. By this way the new application PDB created will have the all the patches applied.

6. Application PDB (entity) configuration from Application Seed
   a. Application PDB is an associated PDB under Application Root. Application PDB will be created by clone from Application Seed.
7. Day Zero Setup

EAR Creation & Deployment

- Co-Deployment – In case of Co-deployment all the product processor objects has to be loaded in the Application Template PDB, which will be cloned into Application Root and then subsequently cloned into Application Seed from Application Root inside an application container. Application Seed is used to accelerate the creation of application PDBs within an application container.
- Stand-alone Deployment– In case of stand-alone deployment, application set up steps has to be followed separately. Installation of multiple product processors can be done inside the same CDB with separate Application containers which has the template PDB, Application Seed and Application PDBs of its own. Same set of installation can be done inside a different CDB.

> **Note**
>
> If .ear deployed in WebLogic server and SOA domain. RCU (Repository Configuration Utility) schemas should be created in a separate PDB. It should not be created within the application root container or in application PDBs.

3.1 Creation of Application Template

3.1.1 Purpose

Application Template PDB is a normal PDB created under CDB to install the required DB objects for a product processor. This PDB will have a common schema and is used as a template for creating Application root through cloning.

3.1.2 Steps to be followed

Below steps to be followed to configure Application Template PDB:

- Application Template PDB Creation
- Property File Creation pointing to Application Template PDB
- Objects loading into the Application Template PDB.

3.1.2.1 Application Template PDB Creation

- User has to login into CDB as a sys user.
- Application Template PDB has to be created under the CDB.
- This Application Template PDB will be kept as a gold copy and recommended to not to use for any other purpose.
- Application Template PDB can have one common schema which will be cloned to create further databases.

Below script will create the Application Template PDB with required grants under the CDB. DBA rights are required to perform this step.

Application Template PDB Creation

(**Refer the Attachment panel of this document to view the script**)

Input sample for the script:

CDB Schema User Name	Sys
CDB Schema Password	Sys
CDB Host	1.1.1.1
CDB Port	1524
CDB Name	FC142CDB
DB Mounted Path	/scratch/db1800dat
Template PDB Name	Templatepdb
Common User Name	CMNUSER
Common User Password	CMNUSER

3.1.2.2 Property file creation with Application Template PDB

- Existing installer will be used for the property file creation.
- Property file has to be created with Application Template PDB schema details. **(Refer OBTF_Installer_Property_File_Creation document)**

3.1.2.3 Loading objects into the Application Template PDB

Objects have to be loaded in the Application Template PDB using bat file [E.g.: SMSDBCompileRun.bat, TFDBCompileRun.bat] by silent installer for respective product processer.

Application Template PDB schema should be checked for sanity with zero invalids.

3.2 Creation of Application Root and Application Seed

- Purpose

- Steps to be followed
  Below script will create the Application root and Application seed. DBA rights are required to perform this step.

3.2.1 Purpose

- Application Root
  - An application root shares some characteristics with the CDB root, because it can contain common objects, and some characteristics with a PDB, because it is created with the CREATE PLUGGABLE DATABASE statement.
- Application Seed
  - After Application Root creation, Application Seed to be created by clone from Application Root. Application seed to be synched with Application Root, whenever there is DB objects deployed in Application Root. i.e., Application seed will have latest DB references of Application Root. Application seed will be used as template to create (entity) Application PDBs.
  - An optional application PDB that serves as a template for creating other PDBs within an application container
- Application Root and Application Seed Creation

3.2.1.1 Application Root and Application Seed Creation

- Application Root
  - Application Root will be created from Application Template PDB through clone. Application Root will hold all the DB objects as single source repository. Initially, the database sources will be copied Application Template PDB. On subsequent patch set upgrade, the database sources will be deployed in Application Root using upgrade mode.
- Application Seed
  - After Application Root creation, Application Seed to be created by clone from Application Root. Application seed to be synched with Application Root, whenever there is DB objects deployed in Application Root. i.e., Application seed will have latest DB references of Application Root. Application seed will be used as template to create (entity) Application PDBs.

3.2.2 Steps to be followed

Below script will create the Application root and Application seed. DBA rights are required to perform this step.

Approot_AppSeed_Creation.sql

(Refer the Attachment panel of this document to view the script)

Input sample for the script:

CDB Schema User Name	Sys
CDB Schema Password	Sys
CDB Host	1.1.1.1

CDB Port	1524
CDB Name	FC142CDB
DB Mounted Path	/scratch/db1800dat
Template PDB Name	Templatepdb
Common User Name	CMNUSER
Common User Password	CMNUSER

3.3 Application Maintenance and PDB creation

3.3.1 Purpose

- Application Maintenance:
  - An application installation is the initial creation of a master application definition. A typical installation creates user accounts, tables, and PL/SQL packages.
- Creation of Application PDB:
  - Application PDB (entity) to be created by clone from Application seed available under Application root. This is associated PDB under Application Root. Any DB sources changes deployed in Application Root, those changes to be synched with Application PDB, if required.
  - Later if new Application PDB to be created, new Application PDB will be created by clone from Application seed. Since Application seed will hold latest DB sources by syncing with Application Root.

3.3.2 Steps for Manual Application Setup

Below steps to be followed to configure Application Root PDB:

- Application Installation
- Application Root objects conversion
- Application Seed Sync with the Application Root
- Creation of Application PDB
- Application Installation
  Application installation has to be done in the approot as version 1.0 with being user made explicit.
- Application Root Objects Conversion
- Application Seed Sync with the Application Root
- Creation of Application PDB
  A PDB that is plugged in to an application container can be created from application seed through cloning.

3.3.2.1 Application Installation

Application installation has to be done in the approot as version 1.0 with being user made explicit.

This application name has to be used for further upgrade in case of object conversion and applying other patch set objects.

Below script will install the application in Application root. DBA rights are required to perform this step.

Application Installation

(Refer the Attachment Panel of this document to view the script)

Input sample for the script:

CDB Schema User Name	Sys
CDB Schema Password	Sys
CDB Host	1.1.1.1
CDB Port	1524
Application Root Name	Approot1
Common User Name	CMNUSER

3.3.2.2 Application Root Objects Conversion

- By default sharing type of all DB objects loaded in the Application Root will be none.
- A static table will hold the information of selected tables for which the sharing type is DATA LINK. Other tables will be treated as METADATA LINK.
- Sharing of object types such as INDEX, LOB, TABLE PARTITION, SEQUENCE, JOB, MATERIALIZED VIEW and DYNAMIC PACKAGES will remain as NONE.
- All other object types such as SYNONYM, VIEW, TRIGGER FUNCTION, PROCEDURE, and PACKAGE would be converted as METADATA LINK.

Object Conversion

- With the above sharing type considerations, DB object types will be converted as DATA LINK and METADATA LINK as part of this application root object conversion step.
- User has to connect to Application Root as common user and then apply changes in upgrade mode with the same application name used in step 3.
- This step will be done from the installer and user will have 4 options to do the conversion as,
- – Shared Application
  – Shared Application and User Authentication
  – Shared Application and Shared Data – Default

- Shared Application and Shared Data – Custom

> **Note**
>
> Application root will be created through cloning from Application Template PDB which will have only the static data. Transaction or maintenance related data will not be available in the Application root.

Shared Application
Here all the function Ids will be available as PDB function ids.

Shared Application and User Authentication
SMS function ids will be available in Approot and the remaining all function ids will be available as PDB function ids.

Shared Application and Shared Data – Default
Identified list of entities will be available in approot and sharing of entities from Approot to individual PDBs is applicable in this model.

Shared Application and Shared Data – Custom
Identified list of entities will be available in approot and sharing of entities from Approot to individual PDBs is applicable in this model.

Additionally, User can opt-out the entities which are not required to be the candidates of approot and those function ids will be moved to PDB.

The application name and type of deployment will be stored in CSTB_PARAM table in approot.

PARAM_NAME	PARAM_VAL
MULTI_TENANT_APP_NAME	OBTFM
MULTI_TENANT_DEPLOYMENT_MODEL	SA (or) SAUA (or) SASDD (or) SASDC

Object conversion is a one-time activity and if it is tried again, system will validate based on the availability of cstb_param values.

3.3.2.3 Application Seed Sync with the Application Root

- In Application Root, post conversion of object type as DATA LINK and METADATA LINK, user need to sync Application Root with Application Seed.
- Post sync, characteristic of objects available in Application seed and Application PDBs will be same.
- Every patch set upgradation in Application Root,
  - User need to sync, Application Root with Application seed, to keep Application seed to hold the latest DB sources since Application seed will be used to create new PDBs further along.
- Below Scripts can also be used to execute this step. This step can be performed from common user.

Approot_AppSeed_Sync.sql

(Refer the Attachment panel of this document to view the script)

Input sample for the script:

Approot Schema Username	CMNUSER
Approot Schema Password	CMNUSER
Approot Host	1.1.1.1
Approot Port	1524
Application Root Name	Approot1
Application Name	FCUBS

3.3.2.4 Creation of Application PDB

A PDB that is plugged in to an application container can be created from application seed through cloning.

Below script will be used to create Application PDB from Application Seed. DBA rights are required to perform this step.

Application PDB Creation

(Refer the Attachment panel of this document to view the script)

Input sample for the script:

CDB Name	FC142CDB
CDB Schema User Name	Sys
CDB Schema Password	Sys
CDB HOST	1.1.1.1
CDB PORT	1522
CDB Mounted Path	/scratch/db1800dat/FC142CDB/templatePDB/users01.dbf
Application Root Name	FCAPPROOT
Application PDB Name	FCAPPPDB1

3.3.3 Steps for Application Setup When Transaction Data Exists

If the maintenance/ transaction data import has to be carried out in the Application root and Application PDBs, below steps has to be followed in the sequential order:

- Creation of Application PDB
- Application Installation
- Application Root objects conversion
- Application PDB Sync with the Application Root
- Application Seed Sync with the Application Root

- Creation of Application PDB
  A PDB that is plugged in to an application container can be created from application seed through cloning.
- Application Installation
  Application installation has to be done in the approot as version 1.0 with being user made explicit.
- Application Root Objects Conversion
- Application PDB Sync with the Application Root
  This topic describes about application PDB sync with the application root.
- Application Seed Sync with the Application Root
  This topic describes about application seed sync with the application root.

3.3.3.1 Creation of Application PDB

A PDB that is plugged in to an application container can be created from application seed through cloning.

Below script will be used to create Application PDB from Application Seed. DBA rights are required to perform this step.

Application_PDB_Creation

(Refer the Attachment panel of this document to view the script)

Input sample for the script:

CDB Name	FC142CDB
CDB Schema User Name	Sys
CDB Schema Password	Sys
CDB HOST	1.1.1.1
CDB PORT	1522
CDB Mounted Path	/scratch/db1800dat
Application Root Name	FCAPPROOT
Application PDB Name	FCAPPPDB1

Note for Shared Application and User Authentication deployment model before object conversion:

SMS function ids will be available in Approot and the remaining all function ids will be available as PDB function ids.

1. Application root before object conversion will only have the static data.
2. If the data import has to be done to the application root schema, following steps 3 to 8 has to be carried out.
3. Triggers have to be disabled in the respective schemas before initiating the import.
4. Tables which are going to be available in the Application root as part of this model can be identified with the below query. (Total of around 21 tables)

```
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 b.object_type = 'TABLE')
AND EXISTS (SELECT 1
FROM cstm_approot_functions_menu c
WHERE c.function_id = a.function_id
AND c.modifiable = 'S');
```

5. The export data dump taken from the entities has to be imported into the application root schema only for these above set of tables.
6. For the PDB's, data from the entities can be directly imported into the respective application PDBs.
7. Once the import is completed, triggers have to be enabled again in the schemas.
8. After the data import, object conversion will be done from the installer.

Example

If there are two entity schemas available for India and Japan region and we have two export dump taken for these schemas.

Step 1: Importing data into the Application root schema
Import the dump taken from India entity schema for the given list of tables followed by the import of dump from Japan entity schema for the same list of tables.

If the table is already present in the application root schema, action should be allowed to just append the table data.

```
impdp Approot_user/Approot_pwd@Approot_Schema
tables= < Tables from the above script>
content=DATA_ONLY
DIRECTORY=DUMP_FC144ENTITY1
DUMPFILE=FC144DEVPDB1_FULDUMP_210519.DMP
LOGFILE=FC144DEVPDB1_FULDUMP_APPROOT_260919_LOG.LOG
REMAP_SCHEMA=FC143ITR:FC14419CM1
REMAP_TABLESPACE=FC143ITR:FC14419CM1
DATA_OPTIONS=skip_constraint_errors
table_exists_action=append transform=OID:n
```

> **Note**
>
> Remap Tablespace recheck in target schema before providing.

Step 2: Importing data into the Application PDB schema
Once the first Application PDB is created from the application seed which will have only the data for static INCs, import the full dump taken from India entity schema.

Similarly, for the second application PDB import the full dump taken from Japan entity schema.

If the table is already present in the application PDB, action should be allowed to just append the table data.

```
impdp Approot_user/Approot_pwd@Approot_Schema
DIRECTORY=DUMP_FC144ENTITY1
DUMPFILE=FC144DEVPDB1_FULDUMP_210519.DMP
LOGFILE=FC144DEVPDB1_FULDUMP_PDB_260919_LOG.LOG
REMAP_SCHEMA=FC143ITR:FC14419CM1
REMAP_TABLESPACE=FC143ITR:FC14419CM1
DATA_OPTIONS=skip_constraint_errors
table_exists_action=append transform=OID:n
```

> **Note**
>
> Remap Tablespace recheck in target schema before providing.

Note for Shared Application and Shared Data – Default deployment model before object conversion:
Identified list of entities will be available in approot and sharing of entities from Approot to individual PDBs is applicable in this model.

1. Application root before object conversion will only have the static data.
2. If the data import has to be done to the application root/ schema, following steps 3 to 8 has to be carried out.
3. Triggers have to be disabled in the respective schemas before initiating the import.
4. Tables which are going to be available in the Application root as part of this model can be identified with the below query. (Total of around 464 tables)

```
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 b.object_type = 'TABLE')
AND EXISTS (SELECT 1
FROM cstm_approot_functions_menu c
WHERE (c.function_id = a.function_id OR

a.function_id IN ('STATIC', 'DYNAMIC')));
```

5. The export data dump taken from the entities has to be imported into the application root schema only for these above set of tables.
6. For the PDB's, data from the entities can be directly imported into the respective application PDBs.

7. Once the import is completed, triggers have to be enabled again in the schemas.
8. After the data import, object conversion will be done from the installer.

Example

If there are two entity schemas available for India and Japan region and we have two export dump taken for these schemas.

Step 1: Importing data into the Application root schema
Import the dump taken from India entity schema for the given list of tables followed by the import of dump from Japan entity schema for the same list of tables.

If the table is already present in the application root schema, action should be allowed to just append the table data.

```
impdp Approot_user/Approot_pwd@Approot_Schema
DIRECTORY=DUMP_FC144ENTITY1
DUMPFILE=FC144DEVPDB1_FULDUMP_210519.DMP
LOGFILE=FC144DEVPDB1_FULDUMP_PDB_260919_LOG.LOG
REMAP_SCHEMA=FC143ITR:FC14419CM1
REMAP_TABLESPACE=FC143ITR:FC14419CM1/USERS
DATA_OPTIONS=skip_constraint_errors
table_exists_action=append transform=OID:n
```

> **Note**
>
> Remap Tablespace recheck in target schema before providing.

Note for Shared Application and Shared Data – Default deployment model before object conversion:
Identified list of entities will be available in approot and sharing of entities from Approot to individual PDBs is applicable in this model.

1. Application root before object conversion will only have the static data.
2. If the data import has to be done to the application root/ schema, following steps 3 to 8 has to be carried out.
3. Triggers have to be disabled in the respective schemas before initiating the import.
4. Tables which are going to be available in the Application root as part of this model can be identified with the below query. (Total of around 464 tables)

```
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 b.object_type = 'TABLE')
AND EXISTS (SELECT 1
FROM cstm_approot_functions_menu c
WHERE (c.function_id = a.function_id OR

a.function_id IN ('STATIC', 'DYNAMIC')));
```

5. The export data dump taken from the entities has to be imported into the application root schema only for these above set of tables.
6. For the PDB's, data from the entities can be directly imported into the respective application PDBs.
7. Once the import is completed, triggers have to be enabled again in the schemas.
8. After the data import, object conversion will be done from the installer.

Example

If there are two entity schemas available for India and Japan region and we have two export dump taken for these schemas.

Step 1: Importing data into the Application root schema
Import the dump taken from India entity schema for the given list of tables followed by the import of dump from Japan entity schema for the same list of tables.

If the table is already present in the application root schema, action should be allowed to just append the table data.

impdp Approot_user/Approot_pwd@Approot_Schema
tables= < **Tables from the above script**>
content=**DATA_ONLY**
DIRECTORY=DUMP_FC144ENTITY1
DUMPFILE=FC144DEVPDB1_FULDUMP_210519.DMP
LOGFILE=FC144DEVPDB1_FULDUMP_APPROOT_260919_LOG.LOG
REMAP_SCHEMA=FC143ITR:FC14419CM1
REMAP_TABLESPACE=FC143ITR:FC14419CM1
DATA_OPTIONS=skip_constraint_errors
table_exists_action=**append** transform=OID:n

> **Note**
>
> Remap Tablespace recheck in target schema before providing.

Step 2: Importing data into the Application PDB schema
Once the first Application PDB is created from the application seed which will have only the data for static INCs, import the full dump taken from India entity schema

Similarly, for the second application PDB import the full dump taken from Japan entity schema.

If the table is already present in the application PDB, action should be allowed to just append the table data.

```
impdp Approot_user/Approot_pwd@Approot_Schema
DIRECTORY=DUMP_FC144ENTITY1
DUMPFILE=FC144DEVPDB1_FULDUMP_210519.DMP
LOGFILE=FC144DEVPDB1_FULDUMP_PDB_260919_LOG.LOG
REMAP_SCHEMA=FC143ITR:FC14419CM1
REMAP_TABLESPACE=FC143ITR:FC14419CM1
DATA_OPTIONS=skip_constraint_errors
table_exists_action=append transform=OID:n
```

> **Note**
>
> Remap Tablespace recheck in target schema before providing.

3.3.3.2 Application Installation

Application installation has to be done in the approot as version 1.0 with being user made explicit.

This application name has to be used for further upgrade in case of object conversion and applying other patch set objects.

Below script will install the application in Application root. DBA rights are required to perform this step.

Application_Installation

(Refer the Attachment Panel of this document to view the script)

Input sample for the script:

CDB Schema User Name	Sys
CDB Schema Password	Sys
CDB HOST	1.1.1.1
CDB PORT	1524
Application Root Name	Approot1
Application PDB Name	FCAPPPDB1
Common User Name	CMNUSER

3.3.3.3 Application Root Objects Conversion

- By default sharing type of all DB objects loaded in the Application Root will be none.

- A static table will hold the information of selected tables for which the sharing type is DATA LINK. Other tables will be treated as METADATA LINK
- Sharing of object types such as INDEX, LOB, TABLE PARTITION, SEQUENCE, JOB, MATERIALIZED VIEW and DYNAMIC PACKAGES will remain as NONE.
- All other object types such as SYNONYM, VIEW, TRIGGER FUNCTION, PROCEDURE, and PACKAGE would be converted as METADATA LINK.

Object Conversion

- With the above sharing type considerations, DB object types will be converted as DATA LINK and METADATA LINK as part of this application root object conversion step.
- User has to connect to Application Root as common user and then apply changes in upgrade mode with the same application name used in step 3.
- This step will be done from the installer and user will have 4 options to do the conversion as,
  - Shared Application
  - Shared Application and User Authentication
  - Shared Application and Shared Data – Default
  - Shared Application and Shared Data – Custom

> **Note**
>
> Application root will be created through cloning from Application Template PDB which will have only the static data. Transaction or maintenance related data will not be available in the Application root.

Shared Application
Here all the function Ids will be available as PDB function ids.

Shared Application and User Authentication

Shared Application and Shared Data – Default
SMS function ids will be available in Approot and the remaining all function ids will be available as PDB function ids.

Shared Application and Shared Data – Default
Identified list of entities will be available in approot and sharing of entities from Approot to individual PDBs is applicable in this model.

Shared Application and Shared Data – Custom
Identified list of entities will be available in approot and sharing of entities from Approot to individual PDBs is applicable in this model.

Additionally, User can opt-out the entities which are not required to be the candidates of approot and those function ids will be moved to PDB.

The application name and type of deployment will be stored in CSTB_PARAM table in approot.

PARAM_NAME	PARAM_VAL
MULTI_TENANT_APP_NAME	OBTFM

PARAM_NAME	PARAM_VAL
MULTI_TENANT_DEPLOYMENT_MODEL	SA (or) SAUA (or) SASDD (or) SASDC

Object conversion is a one-time activity and if it is tried again, system will validate based on the availability of cstb_param values.

3.3.3.4 Application PDB Sync with the Application Root

This topic describes about application PDB sync with the application root.

- In Application Root, post conversion of object type as DATA LINK and METADATA LINK, user need to sync Application PDB with Application Root.
- Post sync, characteristic of objects available in Application root and Application PDBs will be the same.
  Below Script can be used to execute this step. This step can be performed from common user.

  Approot AppSeed_Sync

  (Refer the Attachment panel of this document to view the script)

  Input sample for the script:

PDB Schema User Name	CMNUSER
PDB Schema Password	CMNUSER
PDB HOST	1.1.1.1
PDB PORT	1524
PDB Name	Approot1
Application Name	FCUBS

3.3.3.5 Application Seed Sync with the Application Root

This topic describes about application seed sync with the application root.

- In Application Root, post conversion of object type as DATA LINK and METADATA LINK, user need to sync Application Root with Application Seed.
- Post sync, characteristic of objects available in Application seed and Application root will be same.
- On every patch set upgrade in Application Root, user need to sync the application root with application seed, to keep Application seed hold the latest DB sources since Application seed will be used to create new PDBs further along.
  Below Scripts can also be used to execute this step. This step can be performed from common user.

  Approot_AppSeed_Sync

  (Refer the Attachment panel of this document to view the script)

  Input sample for the script:

Approot Schema User Name	CMNUSER

Approot Schema Password	CMNUSER
Approot HOST	1.1.1.1
Approot PORT	1524
Approot Name	Approot1
Application Name	FCUBS

3.4 Day Zero Setup

Existing Installer can be used to do day zero setup with configuration mode as ‘Application Root’ and by selecting the radio button ‘Utilities’. This step has to be executed for every entity PDB separately.

(Refer: **OBTF_DB_Setup document)**

3.5 EAR Creation and Deployment

- Existing installer file TFEarRun.bat can be used to create EAR.
- EAR deployment has to be deployed manually from console. During EAR deployment, JNDI connectivity details to be maintained for every Application PDB. JNDI details of Application PDB will be captured during Day Zero Setup.

4 EAR Creation and Deployment

- Existing installer file TFEarRun.bat can be used to create EAR.
- EAR deployment has to be deployed manually from console. During EAR deployment, JNDI connectivity details to be maintained for every Application PDB. JNDI details of Application PDB will be captured during Day Zero Setup.
- Approot Object Conversion: Shared Application
  This topic provides systematic instructions about approot object conversion - shared Application.
- Approot Object Conversion: Shared Application and User Authentication
  This topic provides systematic instructions about approot object conversion - shared application and user authentication.
- Approot Object Conversion: Shared Application and Shared Data – Default
  This topic provides systematic instructions about approot object conversion - shared application and shared data – default.
- Approot Object Conversion: Shared Application and Shared Data – Custom
  This topic provides systematic instructions about approot object conversion - shared application and shared data – default.

4.1 Approot Object Conversion: Shared Application

This topic provides systematic instructions about approot object conversion - shared Application.

Shared Application

Kindly make sure all dynamic package exceptions should have an entry in “CSTM_APPROOT_OBJECTS” table.

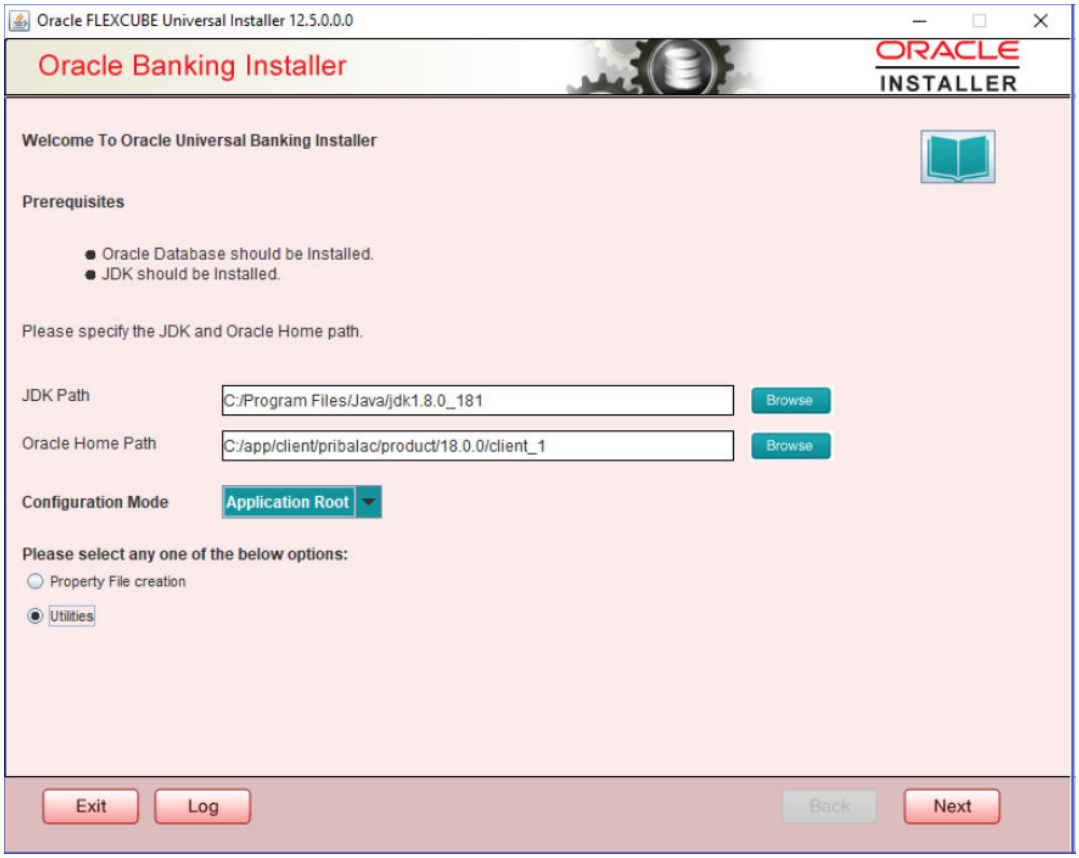
Example: Only package body will be considered as exception and package will be converted to METADATA link.

```
select * from cstm_approot_objects where object_name = 'CLPKS_MSG_SHELL';
```

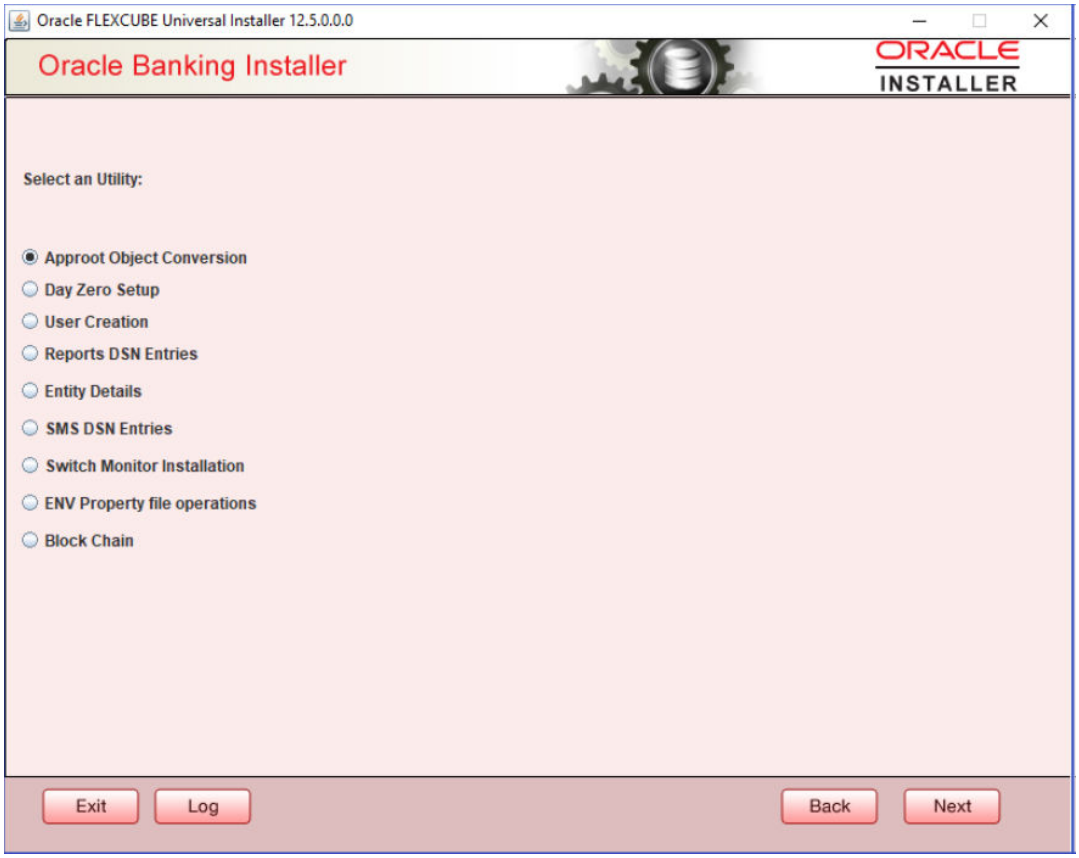
	FUNCTION_ID	OBJECT_NAME	OBJECT_TYPE	SHARING
1	DYNAMIC	CLPKS_MSG_SHELL	PACKAGE BODY	NONE

For multi-tenant deployment setup using the installer with deployment model as ‘Shared Application’, follow the steps given below.

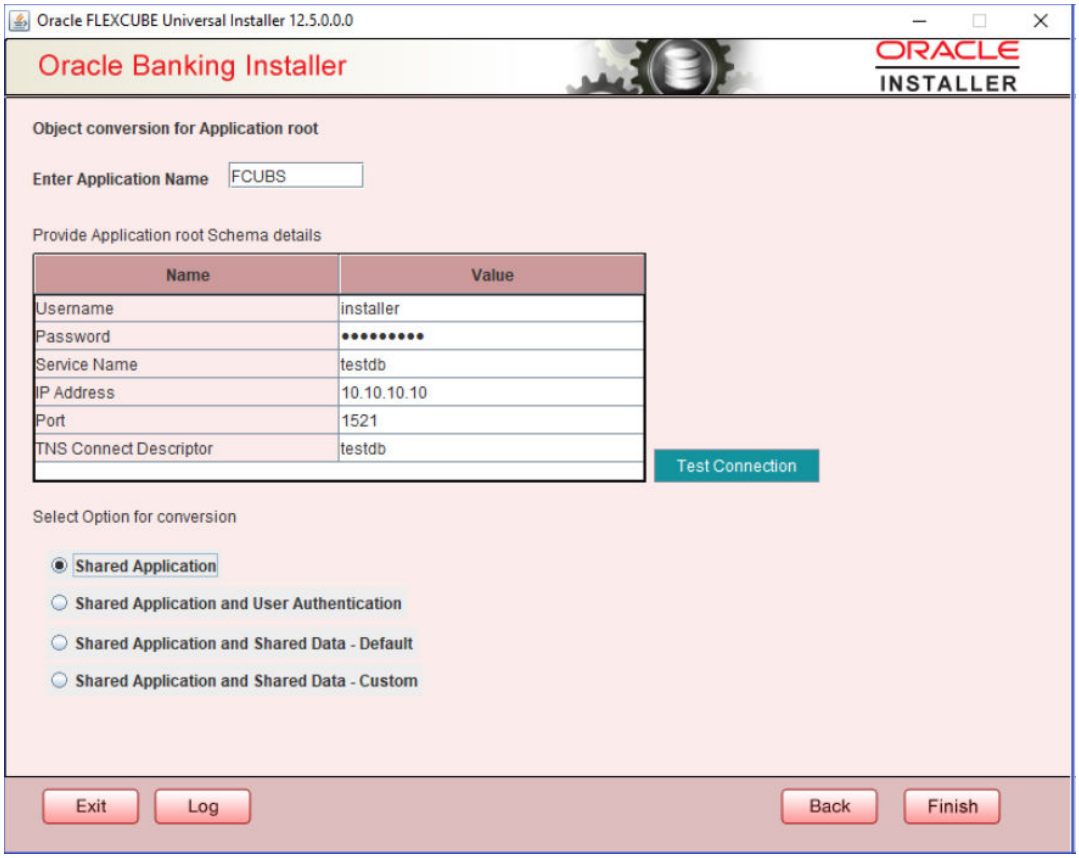
1. Double-click ‘`FCUBSInstaller.bat`’ batch file to launch Oracle FLEXCUBE Universal Installer. The following screen is displayed. Select Utilities option, configuration mode as “**Application Root**” and click ‘**Next**’ button.



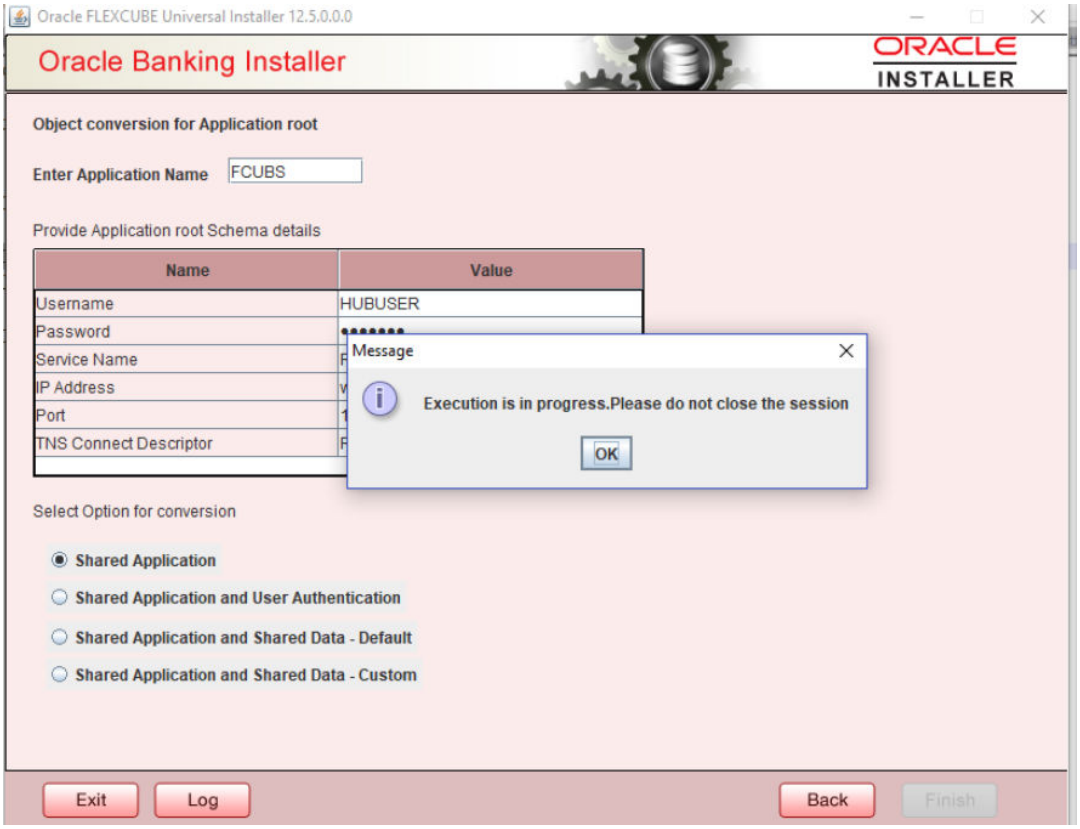
2. Select '**Approot Object Conversion**" in Utility Screen and click **Next** as shown below:



3. In the **Approot Object Conversion** screen, **Enter Application Name** and the Application root schema details where the conversion has to be applied and click on '**Test Connection**'.
4. Once the connection is successful, '**Finish**' button will be enabled.
5. User has to select the option '**Shared Application**' and click on the '**Finish**' button to complete object conversion.



6. Execution will take few minutes and post completion, a dialog box displays '**Compilation Success**' message in the front end.



7. This completes the setup and user can click on **Exit** to close the session.

4.2 Approot Object Conversion: Shared Application and User Authentication

This topic provides systematic instructions about approot object conversion - shared application and user authentication.

Shared Application and User Authentication

Kindly make sure all dynamic package exceptions should have an entry in “CSTM_APPROOT_OBJECTS” table.

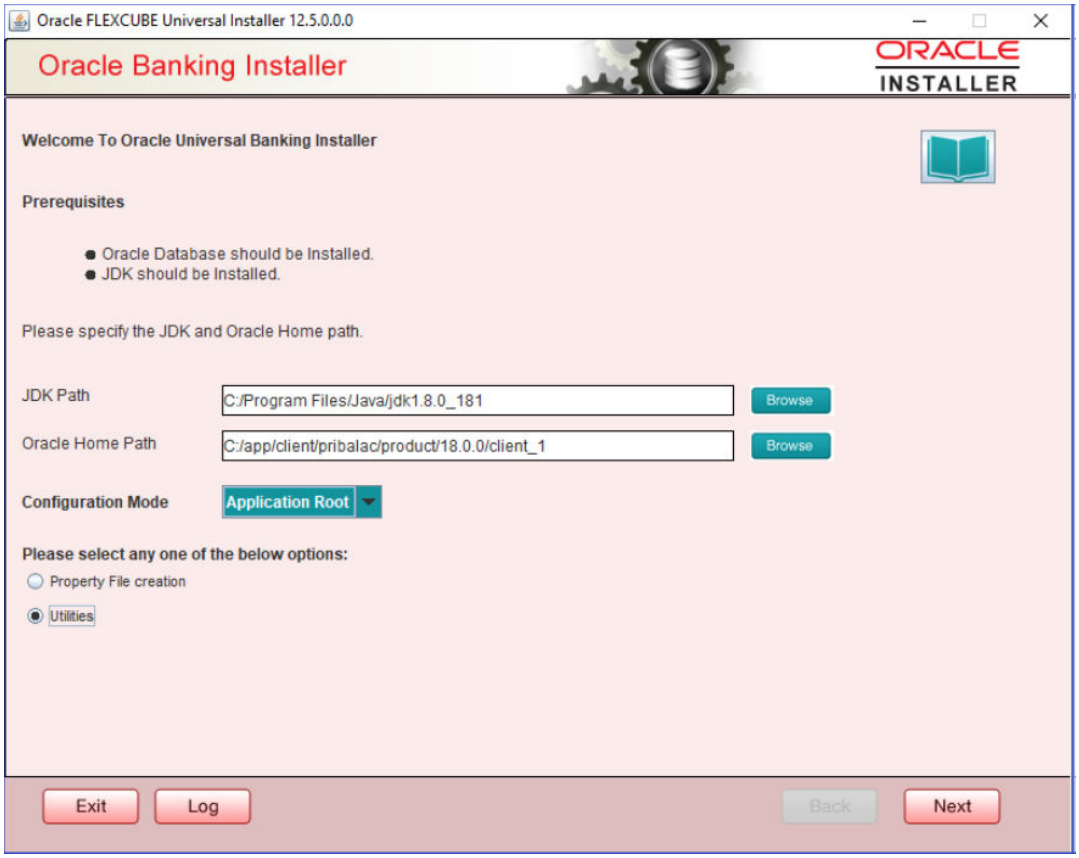
Example: Only package body will be considered as exception and package will be converted to METADATA link

```
select * from cstm_approot_objects where object_name = 'CLPKS_MSG_SHELL';
```

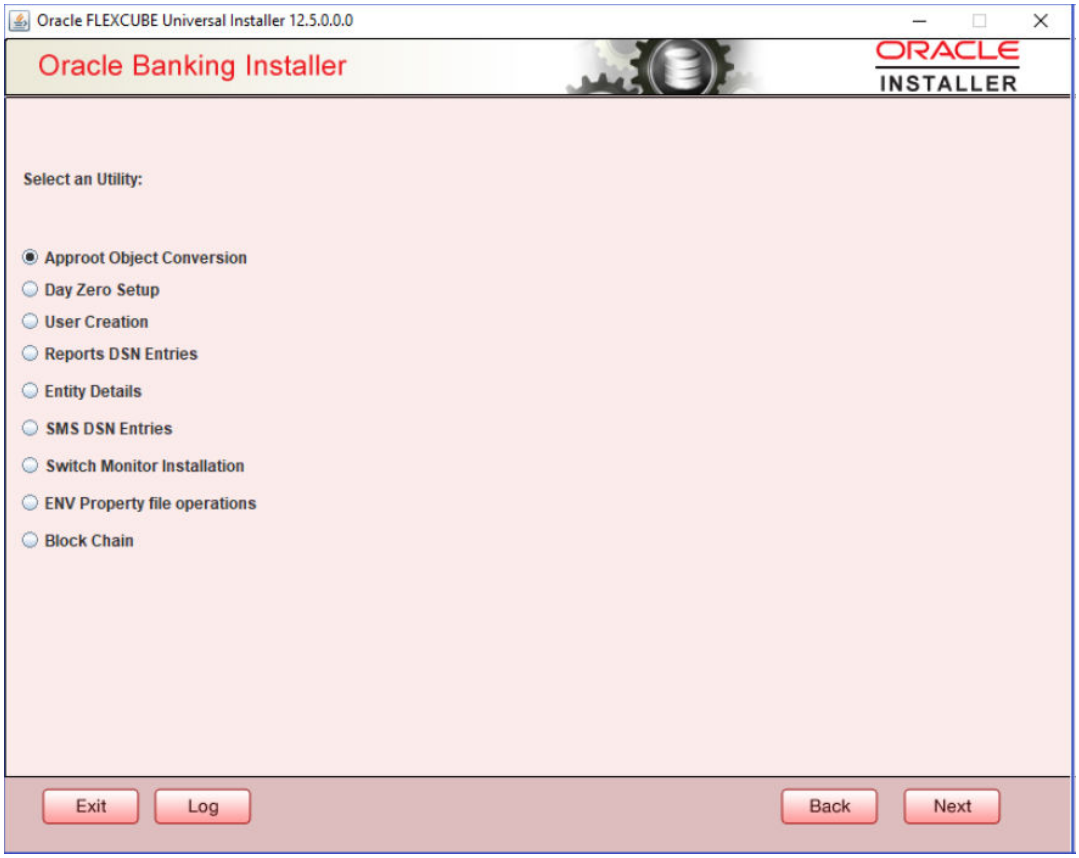
	FUNCTION_ID	OBJECT_NAME	OBJECT_TYPE	SHARING
1	DYNAMIC	CLPKS_MSG_SHELL	PACKAGE BODY	NONE

For multi-tenant deployment setup using the installer with deployment model as ‘Shared Application and User Authentication’, follow the steps given below.

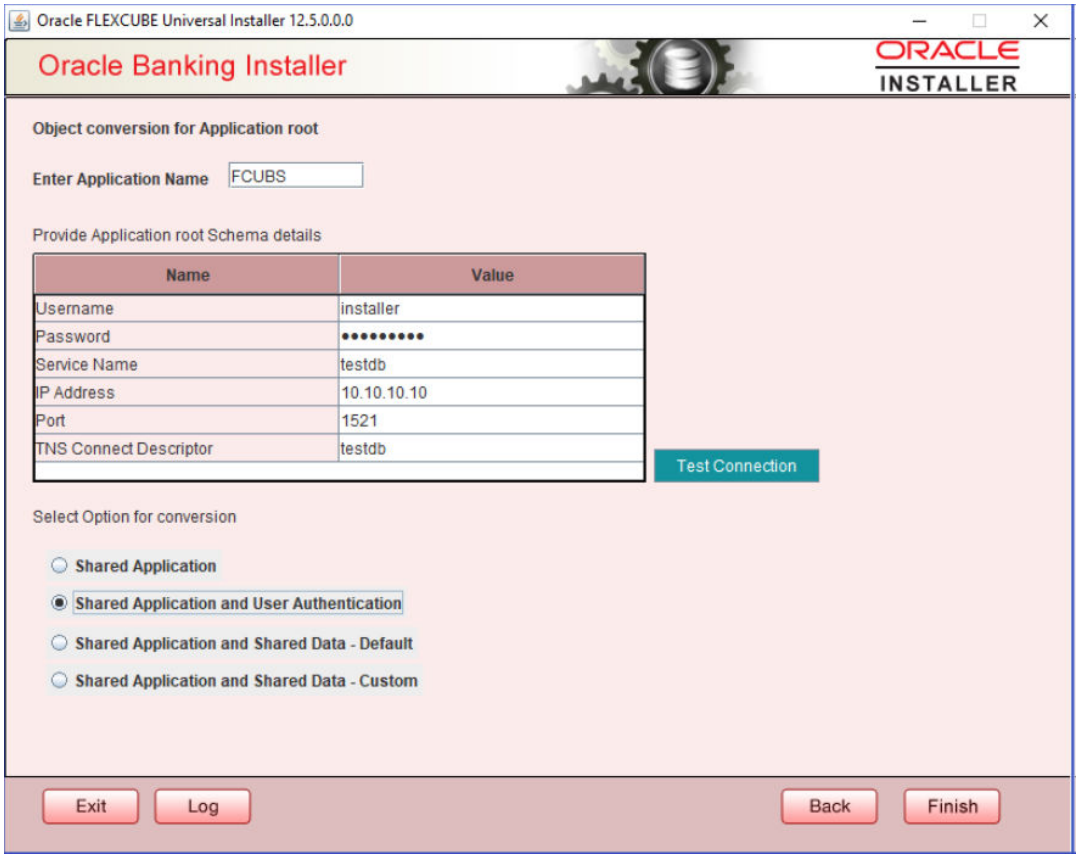
1. Double-click '`FCUBSInstaller.bat`' batch file to launch Oracle FLEXCUBE Universal Installer. The following screen is displayed. Select **Utilities** option, configuration mode as "**Application Root**" and click '**Next**' button.



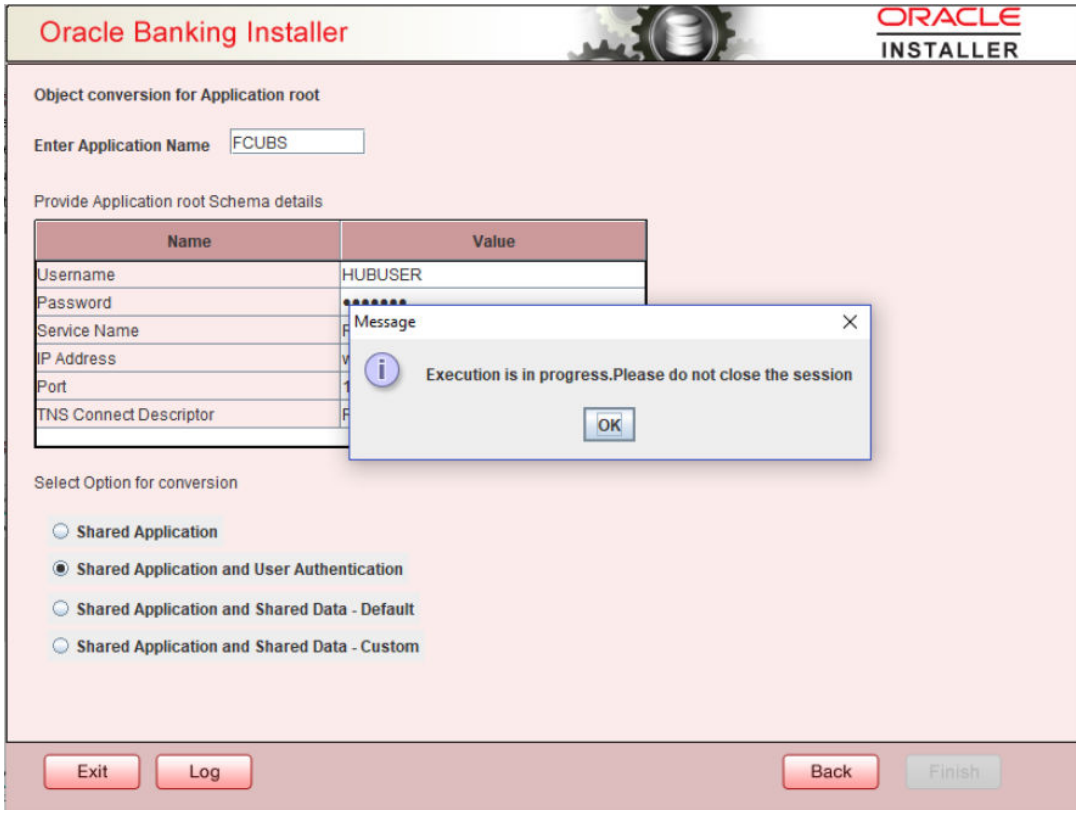
2. Select '**Approot object Conversion**" in Utility Screen and click **Next** as shown below:



3. In the Approot object conversion screen, **Enter Application Name** and the Application Root schema details where the conversion has to be applied and click on '**Test Connection**'.
4. Once the Connection is successful, '**Finish**' button will be enabled.
5. User has to select the option '**Shared Application and User Authentication**' and click on the '**Finish**' button to complete object conversion.



6. Execution will take few minutes and post completion, a dialog box displays '**Compilation Success**' message in the front end.



7. This completes the setup and user can click on **Exit** to close the session.

4.3 Approot Object Conversion: Shared Application and Shared Data – Default

This topic provides systematic instructions about approot object conversion - shared application and shared data – default.

Shared Application and User Authentication - Default

Kindly make sure all dynamic package exceptions should have an entry in “CSTM_APPROOT_OBJECTS” table.

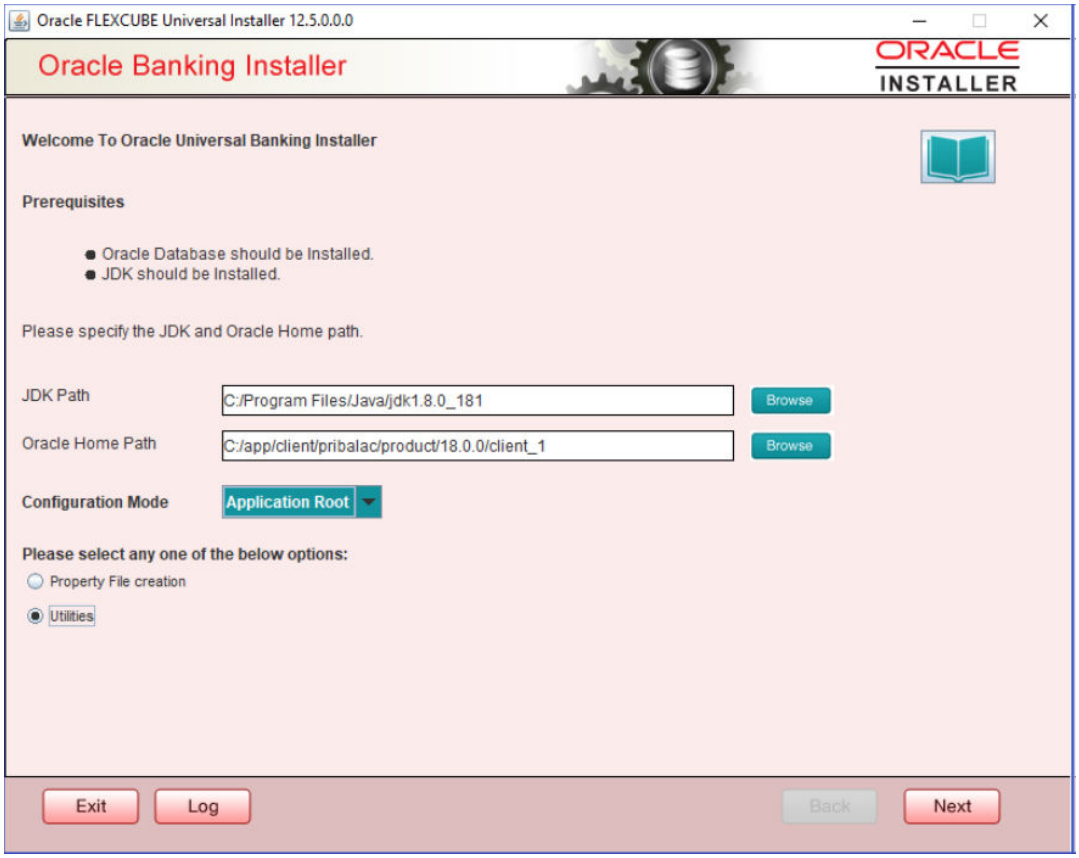
Example: Only package body will be considered as exception and package will be converted to METADATA link

```
select * from cstm_approot_objects where object_name = 'CLPKS_MSG_SHELL';
```

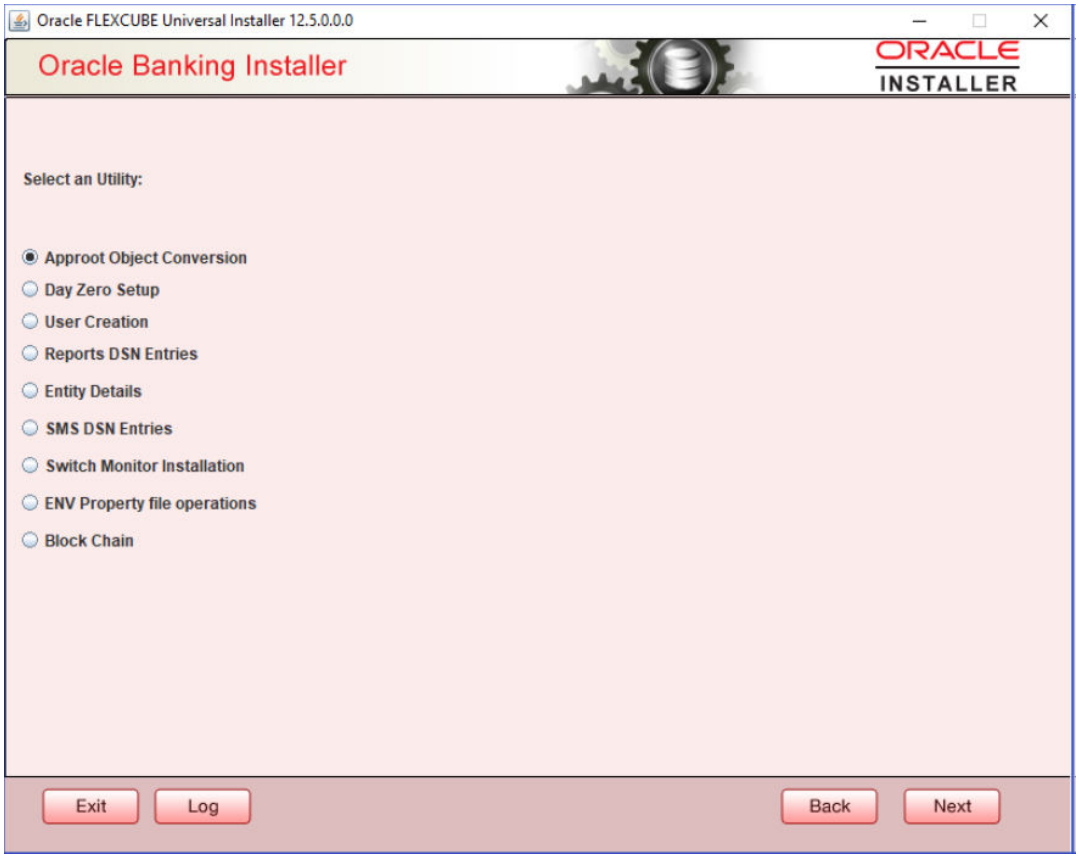
	FUNCTION_ID	OBJECT_NAME	OBJECT_TYPE	SHARING
1	DYNAMIC	CLPKS_MSG_SHELL	PACKAGE BODY	NONE

For multi-tenant deployment setup using the installer with deployment model as ‘Shared Application and Shared Data - Default’, follow the steps given below.

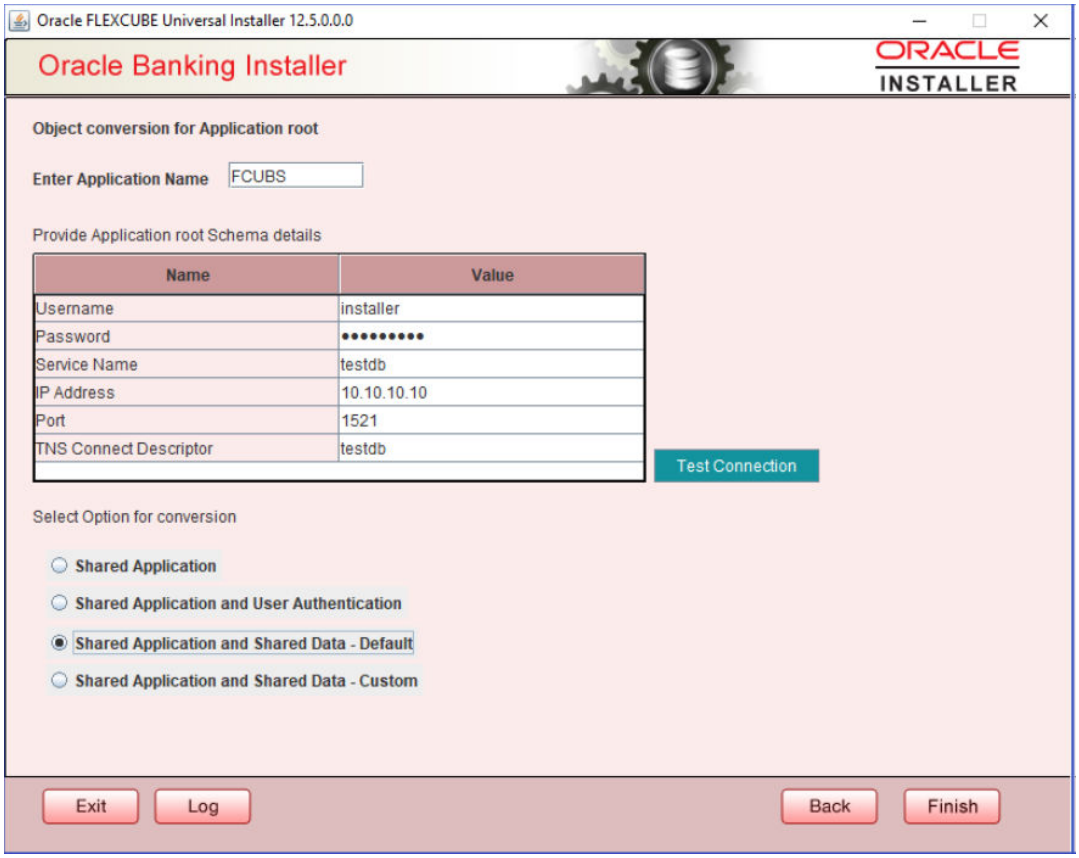
1. Double-click '`FCUBSInstaller.bat`' batch file to launch Oracle FLEXCUBE Universal Installer. The following screen is displayed. Select **Utilities** option, configuration mode as "**Application Root**" and click '**Next**' button.



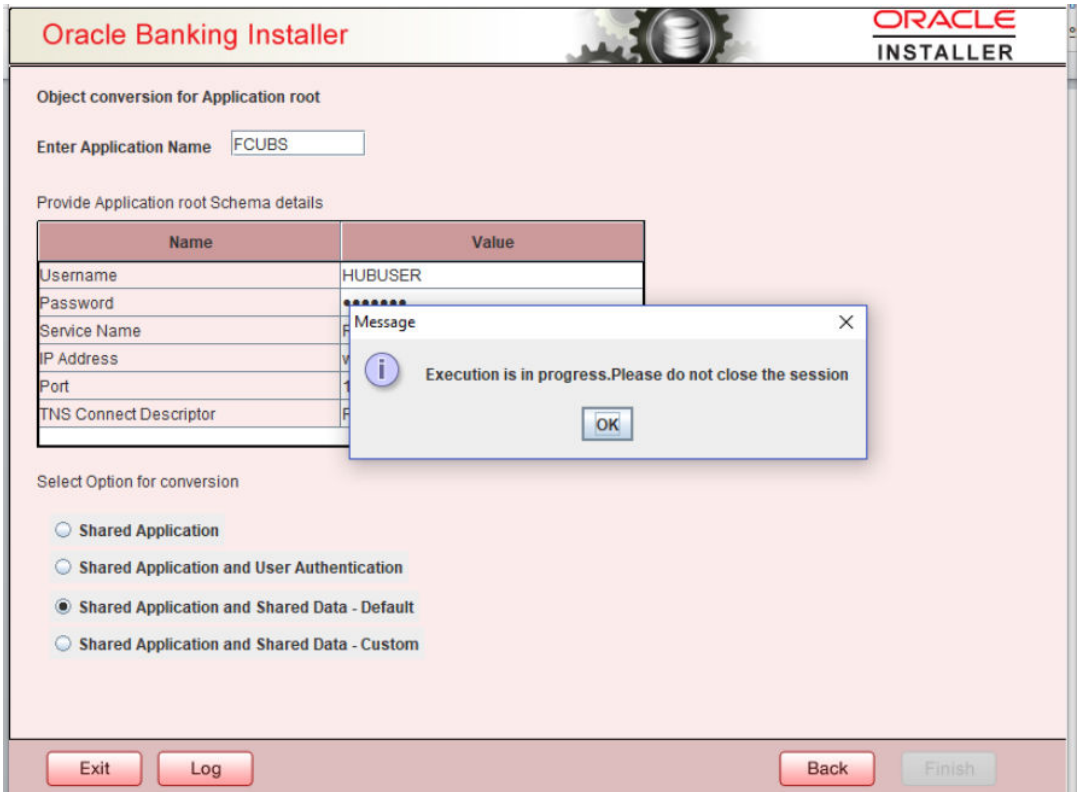
2. Select '**Approot object Conversion**" in Utility Screen and click **Next** as shown below:



3. In the Approot object conversion screen, **Enter Application Name** and the Application Root schema details where the conversion has to be applied and click on '**Test Connection**'.
4. Once the Connection is successful, '**Finish**' button will be enabled.
5. User has to select the option '**Shared Application and Shared Data - Default**' and click on the '**Finish**' button to complete object conversion.



6. Execution will take few minutes and post completion, a dialog box displays '**Compilation Success**' message in the front end.



7. This completes the setup and user can click on **Exit** to close the session.

4.4 Approot Object Conversion: Shared Application and Shared Data – Custom

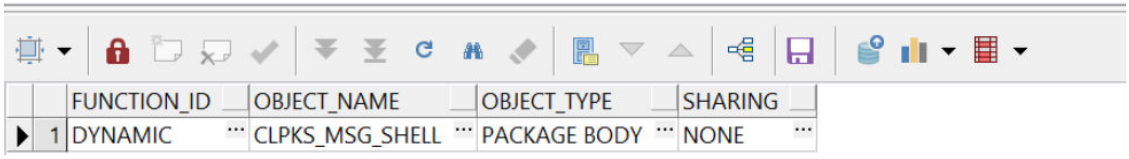
This topic provides systematic instructions about approot object conversion - shared application and shared data – default.

Shared Application and Shared Data – Custom

Kindly make sure all dynamic package exceptions should have an entry in "CSTM_APPROOT_OBJECTS" table.

Example: Only package body will be considered as exception and package will be converted to METADATA link

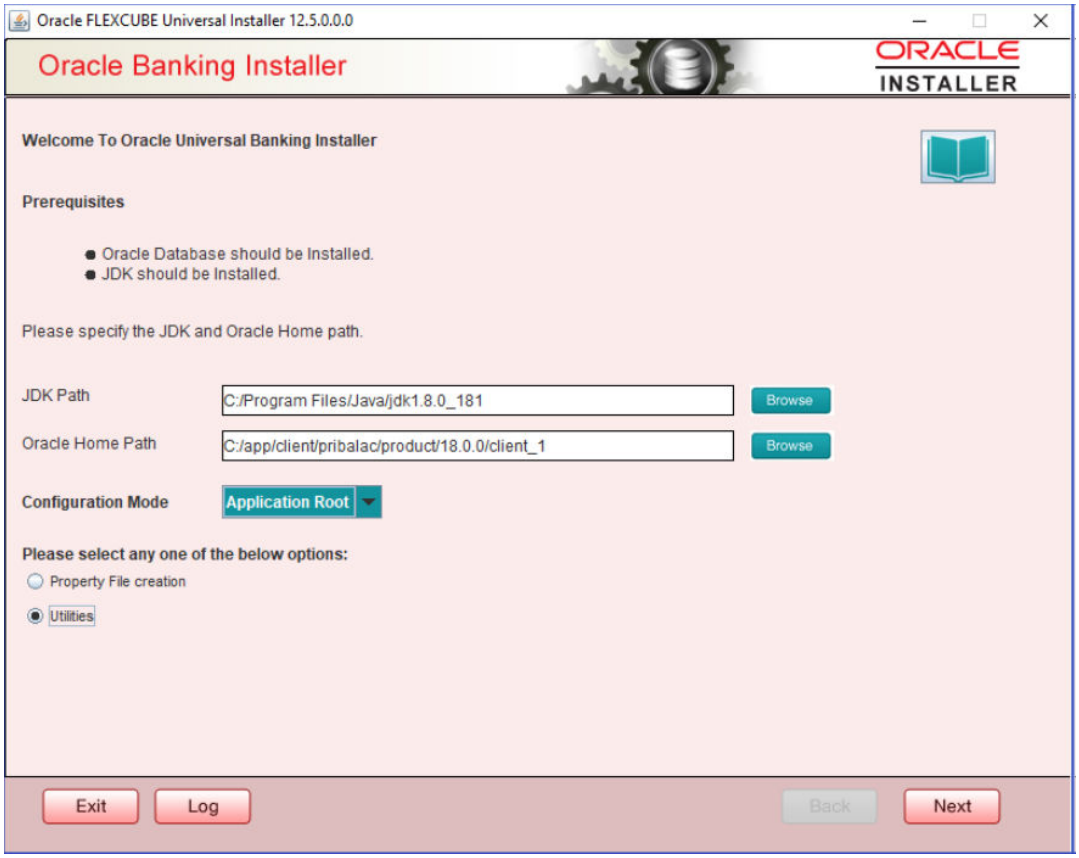
```
select * from cstm_approot_objects where object_name = 'CLPKS_MSG_SHELL';
```



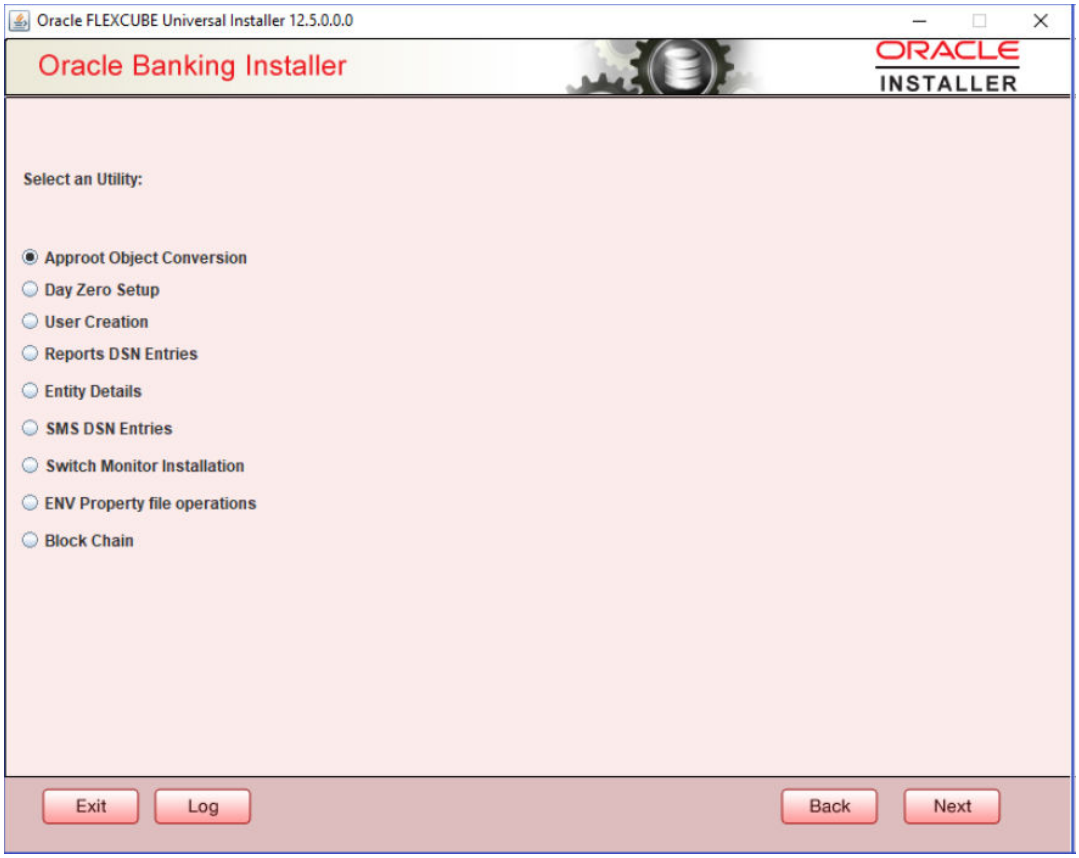
	FUNCTION_ID	OBJECT_NAME	OBJECT_TYPE	SHARING
1	DYNAMIC	CLPKS_MSG_SHELL	PACKAGE BODY	NONE

For multi-tenant deployment setup using the installer with deployment model as 'Shared Application and Shared Data -Custom', follow the steps given below.

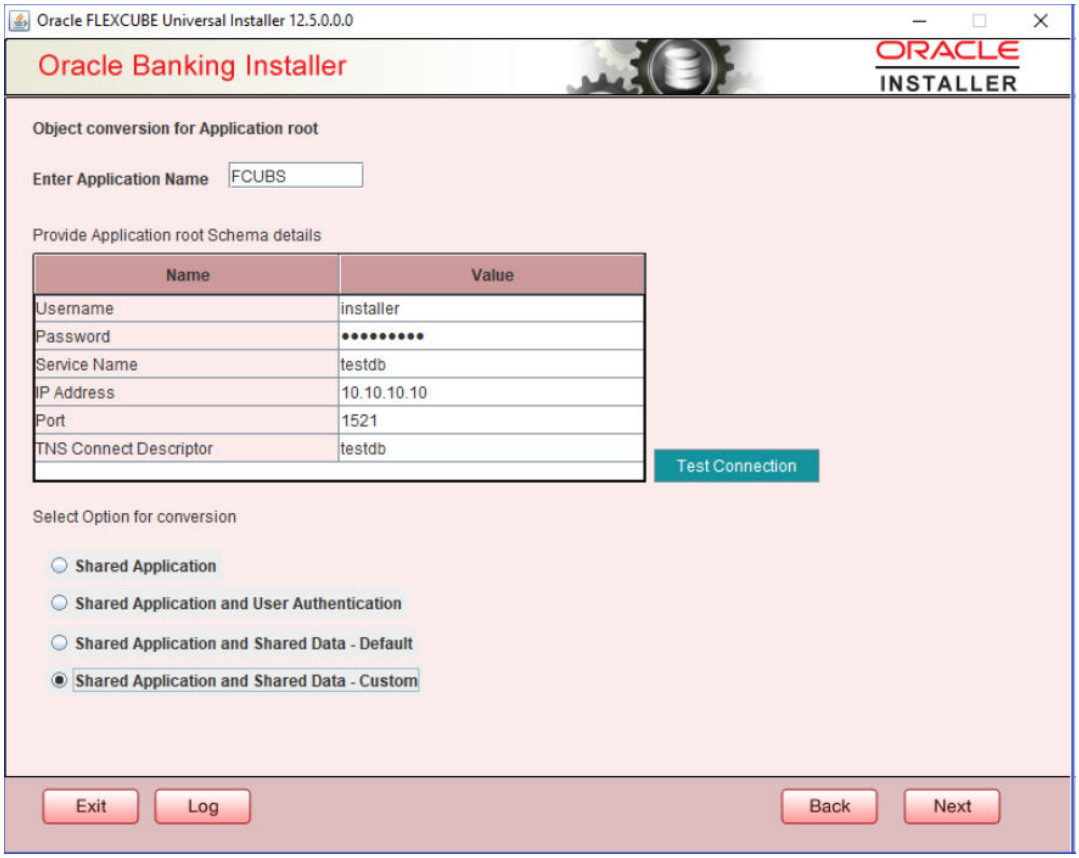
1. Double-click '`FCUBSInstaller.bat`' batch file to launch Oracle FLEXCUBE Universal Installer. The following screen is displayed. Select **Utilities** option, configuration mode as "**Application Root**" and click '**Next**' button.



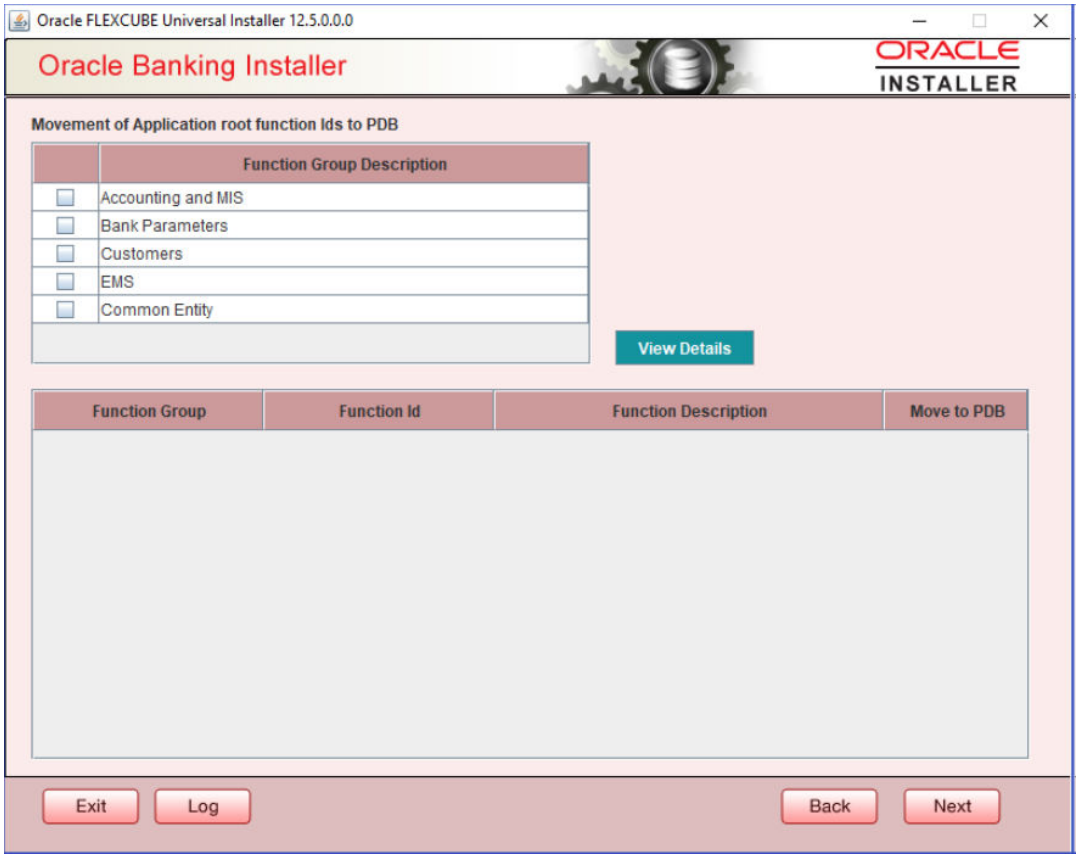
2. Select '**Approot object Conversion**" in Utility Screen and click **Next** as shown below:



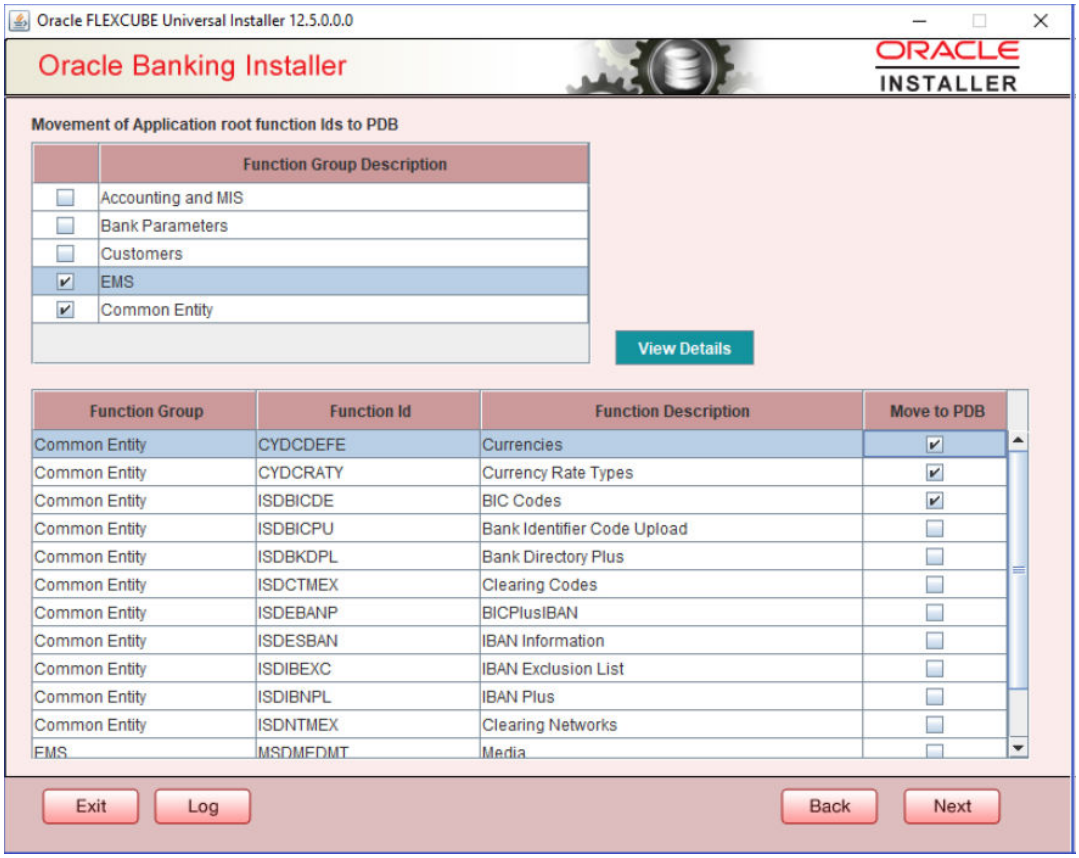
3. In the Approot object conversion screen, **Enter Application Name** and the Application Root schema details where the conversion has to be applied and click on '**Test Connection**'.
4. Once the Connection is successful, '**Next**' button will be enabled.
5. User has to select the option '**Shared Application and Shared Data - Custom**' and click on the '**Next**' button to take through the steps of movement of function ids to PDB.



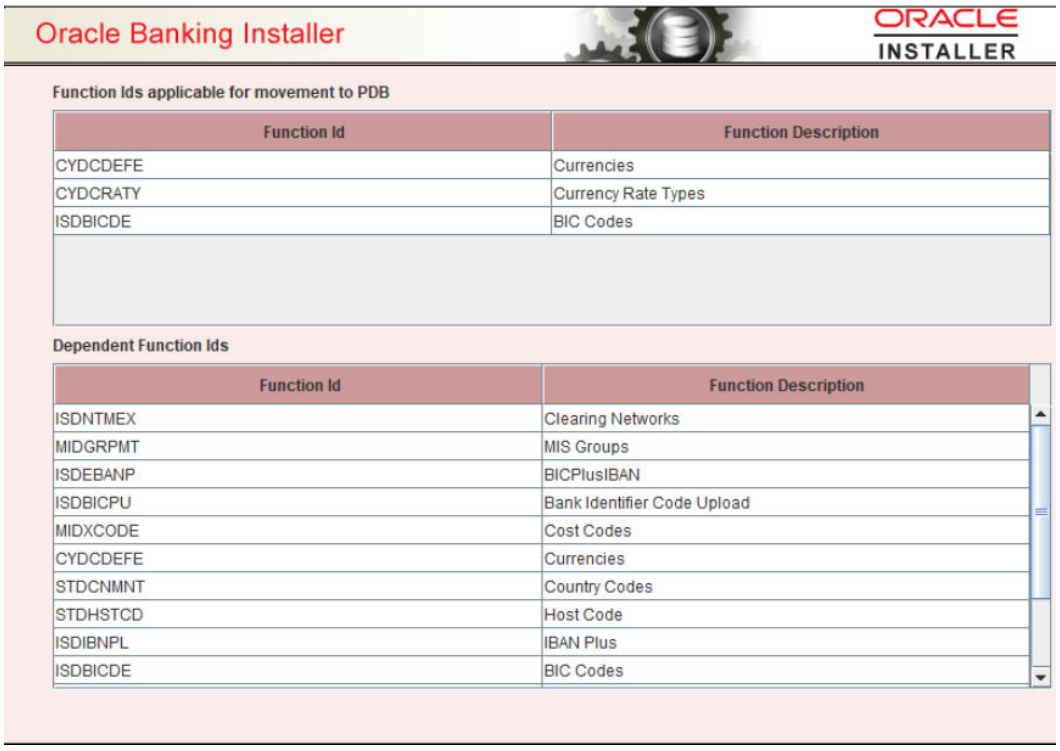
6. In the Next Screen, user can opt-out the entities which are not required to be the candidates of approot and those function ids will be moved to PDB.
7. There will be two multi blocks available.
   a. First multi block will list the details of function groups which are the Approot candidates.
   b. Second multi block will list the function ids corresponding to each of the function group in the first block.
8. User can select more than one function group and the respective function ids will also be appended to the second multi block against the function group on click of '**View Details**' button.



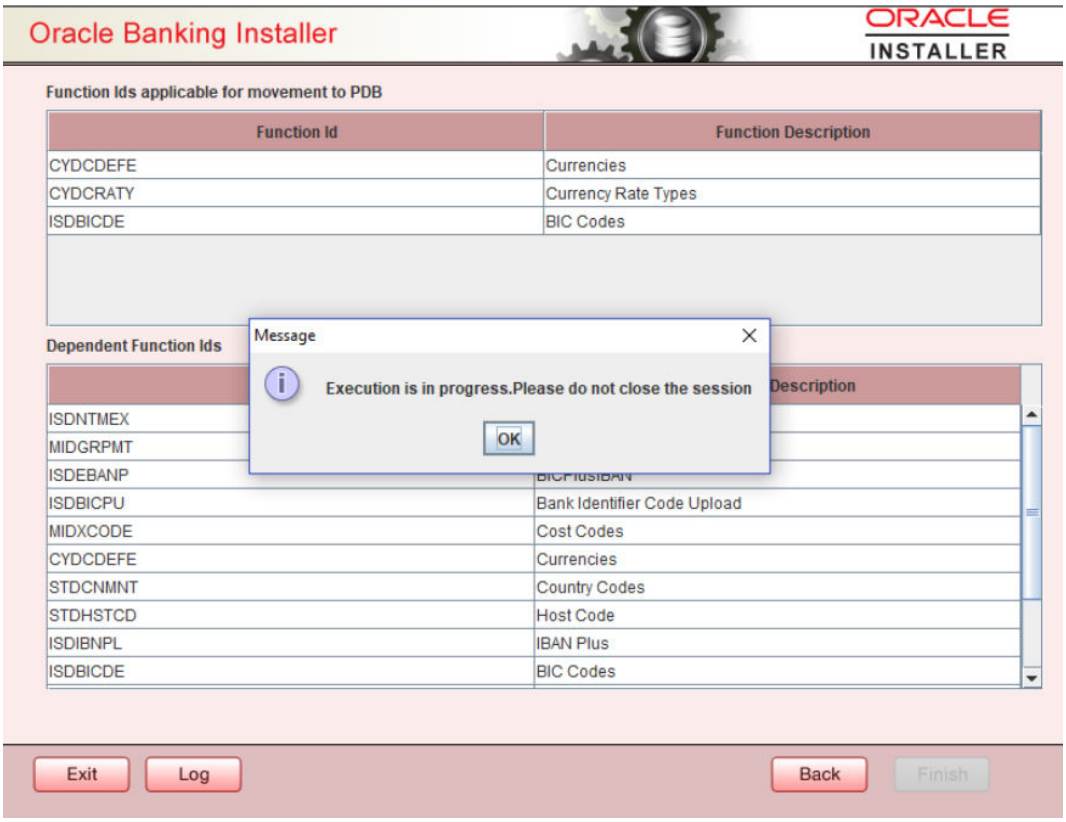
9. Second multi block will have the check box '**Move to PDB**' against each function ID.



10. Once the selection is completed, click on the '**Next**' button to move to the next screen where the complete list of function ids.
11. The dependent function ids of the selected functions opted to move to PDB will be listed in the below section.



12. Object conversion can be completed by clicking on the **Finish** button.
13. Execution will take few minutes and post completion, a dialog box displays '**Compilation Success**' message in the front end.



14. This completes the setup and user can click on **Exit** to close the session.

5

Mandatory step before PDB/SEED Sync

This topic provides systematic instructions to mandatory step before PDB/SEED sync.

1. Login into the Application Entity PDB/SEED as sys user.
2. Create log_error table using Log_Error_Table.DDL followed by create function fn_error_handler.fnc

   Log_Error_Table

   fn_error_handler
3. Alter the DB Syncing error handling parameters.

   Alter the DB Syncing error handling parameters.

   ALTER DATABASE PROPERTY SET SYNC_ERROR_HANDLER = 'sys.fn_error_handler';

   Below are the errors handled during sync in Application PDB / Entity PDB.

Oracle Docs		
Oracle Error	**Cause**	**Action**
ORA-24344	A sql/plsql compilation error occurred.	Return OCI_SUCCESS_WITH_INFO along with the error code.
ORA-06512	Backtrace message as the stack is unwound by unhandled exceptions.	Fix the problem causing the exception or write an exception handler for this condition. Or you may need to contact your application administrator or DBA.
ORA-65297	An operation was attempted that can only be performed outside an application action (install, uninstall, upgrade, or patch)	Perform the operation outside an application action.
ORA-65274	An operation was attempted that can only be performed in an application action (install, uninstall, upgrade, or patch).	Begin an application action.
ORA-00001	An UPDATE or INSERT statement attempted to insert a duplicate key. For Trusted Oracle configured in DBMS MAC mode, you may see this message if a duplicate entry exists at a different level.	Either remove the unique restriction or do not insert the key.
ORA-01430	An ALTER TABLE ADD statement specified the name of a column that is already in the table. All column names must be unique within a table.	Specify a unique name for the new column, then re-execute the statement.

Oracle Docs		
Oracle Error	**Cause**	**Action**
ORA-02264	The specified constraint name has to be unique.	Specify a unique constraint name for the constraint.
ORA-01434	A DROP SYNONYM statement specified a synonym that does not exist. Existing synonym names may be listed by querying the data dictionary.	Specify the name of an existing synonym in the DROP SYNONYM statement.
ORA-00955	An attempt was made to create a database object (such as a table, view, cluster, index, or synonym) that already exists. A user's database objects must have distinct names.	Enter a unique name for the database object or modify or drop the existing object so it can be reused.
ORA-06550	Usually a PL/SQL compilation error.	None
ORA-04063	Cause: Attempt to execute a stored procedure or use a view that has errors. For stored procedures, the problem could be syntax errors or references to other, non-existent procedures. For views, the problem could be a reference in the view's defining query to a non-existent table. Can also be a table which has references to non-existent or inaccessible types.	Fix the errors and/or create referenced objects as necessary.

6 Possible Issues / FAQ

This topic explains about possible issues / FAQ

Significance of the Application Name

The Application name provided at step 3 of the deployment will be used for any object modification like object conversion or patch-set application. Suggested name – FCUBS.

Roles for the Common User

The common user should have DBA role while application install or upgrade. It can be revoked once the application maintenance is completed.

Can there be multiple Applications available in case of Co- deployment?

- It is recommended to have a single application as the Common core units can be released as part of any product processor and if the object can be linked to only one application.
- Modification of the object belonging to one application cannot be modified in another application.

Day zero –set up in multi- tenant

- Day zero set up has be done for each of the PDBs created under the approot. The record insertion will be based on the sharing type of the object.
- If the sharing is METADATA LINK, then the record for the table will be inserted into PDB schema and if the sharing is DATA LINK, record insertion happens in the approot schema for that table.

PDB creation possible errors

Encountered the below error when the template PDB has read only schemas also available additionally.

```
ORA-65005: missing or invalid file name pattern for file - /scratch/db1800dat/BRVCDB19C/SEEDFC142APPROOT/temp012018-01-08_16-05-42-077-PM.dbf
```

In such case, the FILE_NAME_CONVERT has to be provided with the full path till the temp file instead of the Approot and PDB path. Below link is referred to resolve this issue:

https://mosemp.us.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=188548547043444&id=1910646.1&displayIndex=1&_afrWindowMode=0&_adf.ctrl-state=2mboo8is2_4

Sync failure with the PDB

- When synch with PDB fails, there is no definite solution available. Back up of the PDB can be taken before an upgrade and in case of synch failure; new PDB can be created and applied with the backup data.

- Generally, for multi-tenant the recommendation is that objects will be compiled in a normal schema to check the sanity and to make sure the Invalids are zero. Once that is successful, the compilation will be done in Multi-tenant database.

Sync with PDB at different time

- Once the application upgrade is completed in approot, it can be synched up to the PDB. If the PDBs are not synched at the same time, there will be a mismatch between the front end and backend objects.
- In such case when a single PDB is parked for synching afterwards, a separate front URL with backup EAR has to be created to point to the PDB schema.

During patch set deployment encountered below issues during sync into entity pdbs

ORA-21700: object does not exist or is marked for delete

ORA-44201: cursor needs to be reparsed

- Root cause can be traced in DBA_APP_ERRORS / DBA_ERRORS oracle table.
- Execute below command in Approot and Pdb. Consolidate list and create a sql file.

```
SELECT INVALIDOBJECT1
FROM (SELECT 'alter ' || REFERENCED_TYPE || ' ' || REFERENCED_NAME ||
' compile;' INVALIDOBJECT1,
1 INDX
FROM USER_DEPENDENCIES
WHERE NAME IN
(SELECT object_name FROM user_objects WHERE status = 'INVALID')
AND TYPE = 'PACKAGE'
AND REFERENCED_TYPE IN ('PACKAGE', 'PACKAGE BODY')
AND REFERENCED_NAME NOT IN ('STANDARD')
UNION
SELECT 'alter ' || OBJECT_TYPE || ' ' || OBJECT_NAME || ' compile;' INVALIDOBJECT1,
2 INDX
FROM USER_OBJECTS
WHERE OBJECT_NAME IN
(SELECT object_name FROM user_objects WHERE status = 'INVALID')
AND OBJECT_TYPE IN ('PACKAGE')
UNION
SELECT 'alter package ' || OBJECT_NAME || ' compile body;' INVALIDOBJECT1,
3 INDX
FROM USER_OBJECTS
WHERE status = 'INVALID'
AND OBJECT_TYPE IN ('PACKAGE BODY'))
ORDER BY INDX;
```

- Start the upgrade in approot.
- Drop the root cause objects.
- Create the root cause objects.
- Execute the sql file placed in a path.
- End upgrade

- Sync to Entity pdb.
- Verify the result using DBA_APP_ERRORS/ DBA_ERRORS/USER_OBJECTS status = ‘INVALID’.

7

Annexure

This topic contains following sub-topics.

7.1 Default Approot Entities for Common Core

1. Core Entities/Maintenances
    a. Country Code
    b. Host Code & Timezone
    c. Currency
    d. Currency Rate types
    e. Language Code
    f. Rate Code Definition****
2. SMS Entities/Maintenances
    a. Entity Maintenance
    b. User Master (SSD)
    c. Role Master (SSD)
    d. Function Maintenance
    e. PII & Mask Maintenance
    f. SSO Parameters
    g. Hot Keys
    h. Customer Access group
    i. Department Maintenance
3. External Entities
    a. External Chart of Accounts
    b. External Transaction Codes
    c. External Credit Approval
4. MIS and UDF
    a. MIS Class & Codes
    b. MIS Group
    c. MIS Cost Codes

* Islamic Entities wherever applicable
*

d. MIS Pool
e. UDF Definition
f. UDF Function ID Mapping

5. Other Entities
   a. BIC Codes and related maintenances
   b. Process Definition
   c. Amount Text
   d. Media
   e. Gateway Multi-Entity Function Ids**
      i. Upload Source
      ii. External System
      iii. Amendment Maintenance

7.2 Default Approot Entities for OBTFM

1. Core Entities/Maintenances
   a. Chart of Accounts
   b. Revaluation Setup
   c. Transaction Codes
   d. Currency Denominations
   e. Customer Categories, Prefixes, Groups, Ownership, Relation
   f. Issuer Codes
   g. Overrides
2. Subsystem and Classes
   a. Commission, Interest, Charge, Tax Scheme Class
   b. Status Codes
   c. ICCF Rule master**
   d. Tax Rule Master**
   e. Tax Categories, Tax Scheme, Tax Rate codes
   f. Product – UDF Mapping**
   g. Message Types, Media, Locations, SWIFT Tags
3. CASA and TD (Conventional and Islamic****)
   a. Account Class and Account Class Group
   b. Interest and Charge Rule and Product

* New function IDs
* New function IDs
* New function IDs
* New function IDs
* Islamic Entities wherever applicable
*

c. Interest and Charge SDEs
d. Standing Instruction Product
e. Structured Deposit Product
f. PDC Product

4. Retail Lending (Conventional and Islamic[****])
   a. Retail Lending Product and related Maintenances
   b. Leasing Product
   c. Mortgage Product
   d. Microfinance Product
   e. Collection Product
5. Teller
   a. Retail Teller Product
   b. Corporate Teller Product
   c. Utility Payment Product
6. Trade (Conventional and Islamic[****])
   a. Letter of Credit Product and Related Maintenances
   b. Bills and Collection Product and Related Maintenances
7. Treasury (Conventional and Islamic[****])
   a. Foreign Exchange Product and Related Maintenances
   b. Money Market Product and Related Maintenances
   c. Securities Repo Product and Related Maintenances
   d. Corporate Deposit Product and Related Maintenances
   e. Securities Product and Related Maintenances
   f. Derivatives Product and Related Maintenances
8. Other Modules (Conventional and Islamic[****])
   a. Asset Management Fund Product
   b. Fixed Assets Product
   c. Expense Processing Product
   d. Intermediary Product
   e. Retail Bills Product

---

* Islamic Entities wherever applicable
*
* Islamic Entities wherever applicable
*
* Islamic Entities wherever applicable
*
* Islamic Entities wherever applicable
*

Application Installation

Purpose

It is used for application installation.

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 Configuration
SPOOL "&SPOOL_PATH"
/* Inputs are recieved */
/* Connect CDB as sys user */
accept P_CDB_USER Prompt 'Enter CDB Schema Username: '
accept P_CDB_PWD Prompt  'Enter CDB Schema Password: '
accept P_CDB_HOST Prompt 'Enter CDB Schema Host: '
accept P_CDB_PORT Prompt 'Enter CDB Schema Port: '
accept P_APPROOT_NAME Prompt 'Enter Application Root Name: '
accept P_APPLICATION_NAME Prompt   'Enter application name to be installed: '
accept P_COMMON_USER Prompt 'Enter Common User Name: '

/* Connecting to Application Root As SYSDBA*/
conn &P_CDB_USER/&P_CDB_PWD@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)
(HOST=&P_CDB_HOST)(PORT=&P_CDB_PORT)))(CONNECT_DATA=(SERVER=DEDICATED)
(SERVICE_NAME=&P_APPROOT_NAME))) as sysdba;

alter pluggable database application &P_APPLICATION_NAME begin install '1.0';
    exec dbms_pdb.set_user_explicit('&P_COMMON_USER');
alter pluggable database application &P_APPLICATION_NAME end install;

SET ERRORLOGGING OFF
SPOOL OFF
```

Application_PDB_Creation

Purpose

It isused in the application PDB creation.

Syntax

/* Pre-requisites: Step 2 on application root and application seed has to be 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 PDB Configuration SPOOL "&SPOOL_PATH" /* Inputs are recieved */ /* Connect Approot as sys user */ accept P_CDB_USER Prompt 'Enter CDB Username: ' accept P_CDB_PWD Prompt 'Enter CDB Password: ' accept P_CDB_HOST Prompt 'Enter CDB Host: ' accept P_CDB_PORT Prompt 'Enter CDB Port: ' accept P_CDB_NAME Prompt 'Enter CDB Schema Name: ' accept P_DB_MOUNTED_PATH Prompt 'Enter Approot mounted path for approot application seed creation: [Eg: /scratch/db1800dat]' accept P_APPROOT_NAME Prompt 'Enter Application Root Name: ' accept P_APPPDB_NAME Prompt 'Please provide name for Application PDB Name -- Application Root associated PDB: ' /* Connecting to Application Root As SYSDBA*/ conn &P_CDB_USER/ &P_CDB_PWD@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP) (HOST=&P_CDB_HOST)(PORT=&P_CDB_PORT))) (CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=&P_APPROOT_NAME))) as sysdba; /* Creating Application Associated PDB*/ CREATE pluggable database &P_APPPDB_NAME FROM &P_APPROOT_NAME$SEED file_name_convert=('&P_DB_MOUNTED_PATH/&P_CDB_NAME/ SEED&P_APPROOT_NAME/','&P_DB_MOUNTED_PATH/&P_APPROOT_NAME/ &P_APPPDB_NAME/'); ALTER pluggable database &P_APPPDB_NAME OPEN; SET ERRORLOGGING OFF SPOOL OFF

Application_Template_PDB_Creation

Purpose

This script is used for application template PDB creation.

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 Template PDB Configuration
SPOOL "&SPOOL_PATH"
/* CDB sys user name and password to be given */
accept P_CDB_USER Prompt 'Enter CDB Schema Username: [Eg: sys]'
accept P_CDB_PWD Prompt 'Enter CDB Schema Password: [Eg: PASSWORD]'
accept P_CDB_HOST Prompt 'Enter CDB Schema Host: [Eg: fcubs.in.oracle.com]'
accept P_CDB_PORT Prompt 'Enter CDB Schema Port: [Eg: 1521]'
accept P_CDB_NAME Prompt 'Enter CDB Service Name: [Eg: FCUBSCDB]'
accept P_DB_MOUNTED_PATH Prompt 'Enter CDB mounted path: [Eg: /scratch/
db1800dat]'

accept P_APP_TEMPLATE_PDB Prompt 'Enter Name for Application Template PDB to
be created: [Eg: pdbfcubs]'
accept P_COMMON_USER Prompt 'Enter Common Username to be created: [Eg: fcubs]'
accept P_COMMON_USER_PWD Prompt 'Enter Pwd for Common User : [Eg: fcubs]'
accept P_COMMON_TSPACE Prompt 'Enter TableSpace Name :[Eg: fcubs]'

/* Connecting to CDB as sysdba */
CONN &P_CDB_USER/&P_CDB_PWD@&P_CDB_NAME AS sysdba;

create pluggable database &P_APP_TEMPLATE_PDB ADMIN USER sourceadmin
IDENTIFIED BY sourceadmin file_name_convert=('pdbseed','&P_APP_TEMPLATE_PDB');

alter pluggable database &P_APP_TEMPLATE_PDB open;
alter pluggable database &P_APP_TEMPLATE_PDB save state;

/*connecting to template pdb as sysdba */
conn &P_CDB_USER/&P_CDB_PWD@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)
(HOST=&P_CDB_HOST)(PORT=&P_CDB_PORT)))(CONNECT_DATA=(SERVER=DEDICATED)
```

```
(SERVICE_NAME=&P_APP_TEMPLATE_PDB))) as sysdba;
create tablespace &P_COMMON_USER datafile '&P_DB_MOUNTED_PATH/&P_CDB_NAME/
&P_APP_TEMPLATE_PDB/&P_COMMON_TSPACE..dbf' size 100M autoextend on next 10M
maxsize 30000M;

CREATE USER &P_COMMON_USER IDENTIFIED BY &P_COMMON_USER_PWD default
tablespace &P_COMMON_USER quota unlimited on &P_COMMON_USER;

grant execute on dbms_sql to &P_COMMON_USER;
grant execute on dbms_lock to &P_COMMON_USER;
grant execute on dbms_job to &P_COMMON_USER;
grant execute on dbms_alert to &P_COMMON_USER;
grant execute on dbms_refresh to &P_COMMON_USER;
grant execute on dbms_pipe to &P_COMMON_USER;
grant execute on dbms_shared_pool to &P_COMMON_USER;
grant execute on dbms_application_info to &P_COMMON_USER;
grant execute on utl_file to &P_COMMON_USER;
grant select on v_$process to &P_COMMON_USER;
grant select on v_$session to &P_COMMON_USER;
grant select on v_$instance to &P_COMMON_USER;
grant select on v_$timer to &P_COMMON_USER;
grant select on v_$database to &P_COMMON_USER;
grant select on v_$parameter to &P_COMMON_USER;
grant select on v_$nls_parameters to &P_COMMON_USER;
grant select on dba_jobs_running to &P_COMMON_USER;
grant create session to &P_COMMON_USER;
grant create synonym to &P_COMMON_USER;
grant create view to &P_COMMON_USER;
grant create sequence to &P_COMMON_USER;
grant create table to &P_COMMON_USER;
grant create procedure to &P_COMMON_USER;
grant create trigger to &P_COMMON_USER;
grant create type to &P_COMMON_USER;
grant create library to  &P_COMMON_USER;
grant create database link to  &P_COMMON_USER;
grant create any synonym to  &P_COMMON_USER;
grant select on dba_jobs to &P_COMMON_USER;
grant create database link to &P_COMMON_USER;
grant create materialized view to &P_COMMON_USER;
grant execute on dbms_aq to &P_COMMON_USER;
grant execute on dbms_aqadm to &P_COMMON_USER;
grant execute on dbms_job to  &P_COMMON_USER;
grant execute on dbms_lock to  &P_COMMON_USER;
grant execute on dbms_pipe to  &P_COMMON_USER;
grant execute on dbms_refresh to  &P_COMMON_USER;
grant execute on dbms_rls to  &P_COMMON_USER;
create public synonym dbms_shared_pool for sys.dbms_shared_pool;
grant execute on dbms_shared_pool to  &P_COMMON_USER;
grant execute on dbms_sql to  &P_COMMON_USER;
grant execute on utl_file to  &P_COMMON_USER;
grant select on SYS.TRANSPORT_SET_VIOLATIONS to  &P_COMMON_USER;
grant create evaluation context to  &P_COMMON_USER;
grant create rule to  &P_COMMON_USER;
grant create job to  &P_COMMON_USER;
grant create rule set to  &P_COMMON_USER;
grant exp_full_database to  &P_COMMON_USER;
```

```
grant alter tablespace to  &P_COMMON_USER;
grant manage tablespace to  &P_COMMON_USER;
grant execute on DBMS_FILE_TRANSFER to  &P_COMMON_USER;
grant execute on SYS.DBMS_TTS to  &P_COMMON_USER;
grant execute on SYS.DBMS_DATAPUMP to  &P_COMMON_USER;
grant JAVAUSERPRIV to &P_COMMON_USER;
grant execute on dbms_scheduler to  &P_COMMON_USER;
create public synonym UTL_RECOMP for sys.UTL_RECOMP;
grant execute on UTL_RECOMP to &P_COMMON_USER;
grant execute on DBMS_MONITOR to &P_COMMON_USER;
grant select on dba_directories to &P_COMMON_USER;
grant execute on DBMS_CRYPTO to &P_COMMON_USER;
grant select on gv_$session to &P_COMMON_USER;
grant create any directory to &P_COMMON_USER;
grant select on SYS.DBA_SCHEDULER_RUNNING_JOBS to &P_COMMON_USER;
grant execute on sys.dbms_redact to &P_COMMON_USER;
grant SELECT on sys.redaction_policies to &P_COMMON_USER;
grant SELECT on sys.redaction_columns to &P_COMMON_USER;
grant SELECT on sys.redaction_values_for_type_full to &P_COMMON_USER;
grant create session,connect,resource to &P_COMMON_USER;
grant SELECT ON dba_applications to &P_COMMON_USER;
grant SELECT ON dba_app_versions to &P_COMMON_USER;
grant dba to &P_COMMON_USER;

SET ECHO OFF
clear screen
spool off
```

Approot_AppSeed_Sync.sql

Purpose

Application Root - PDB Model Configuration

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 Configuration
SPOOL "&SPOOL_PATH"
/* Inputs are received */
accept P_APPROOT_USER Prompt 'Enter Approot Schema Username: '
accept P_APPROOT_PWD Prompt  'Enter Approot Schema Password: '
accept P_APPROOT_HOST Prompt 'Enter Approot Schema Host: '
accept P_APPROOT_PORT Prompt 'Enter Approot Schema Port: '
accept P_APPROOT_NAME Prompt 'Enter Application Root Name: '

accept P_APPLICATION_NAME Prompt   'Enter application name to be upgraded for
object conversion: '

/*Connecting to Application seed*/
conn &P_APPROOT_USER/
&P_APPROOT_PWD@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)
(HOST=&P_APPROOT_HOST)(PORT=&P_APPROOT_PORT)))(CONNECT_DATA=(SERVER=DEDICATED)
(SERVICE_NAME=&P_APPROOT_NAME$SEED)));

/*Synching object conversion to application seed */
alter pluggable database application &P_APPLICATION_NAME sync;

SET ERRORLOGGING OFF
SPOOL OFF
```

Approot_PDB_Sync

Purpose

Check the approot PDB Sync.

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"
/* Inputs are received */
accept P_PDB_USER Prompt 'Enter PDB Schema Username: '
accept P_PDB_PWD Prompt 'Enter PDB Schema Password: '
accept P_PDB_HOST Prompt 'Enter PDB Schema Host: '
accept P_PDB_PORT Prompt 'Enter PDB Schema Port: '

accept P_PDB_NAME Prompt 'Enter the PDB name to be synched: '
accept P_APPLICATION_NAME Prompt 'Enter the application name: '

/*Connecting to pdb */
conn &P_PDB_USER/&P_PDB_PWD@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)
(HOST=&P_PDB_HOST)(PORT=&P_PDB_PORT)))(CONNECT_DATA=(SERVER=DEDICATED)
(SERVICE_NAME=&P_PDB_NAME)));

/*Synching the application with pdbs */
alter pluggable database application &P_APPLICATION_NAME sync;

SET ERRORLOGGING OFF
SPOOL OFF
```

fn_error_handler

Purpose

This script is used as error handler.

Syntax

```
CREATE OR REPLACE FUNCTION fn_error_handler(octcode   IN NUMBER,
                                            errcode   IN NUMBER,
                                            statement IN VARCHAR2,
                                            resync    IN NUMBER)
  RETURN NUMBER AUTHID CURRENT_USER is
  retcode NUMBER := DBMS_PDB_APP_CON.SYNC_ERROR_NOT_OK;
BEGIN
  IF errcode IN
(24344,6512,65297,65272,65274,4045,1,2264,1430,1434,955,4063,942,4043,65215,22
60,904,4023,6510,4097,6508,4088) THEN
    retcode := DBMS_PDB_APP_CON.SYNC_ERROR_OK_ALWAYS;
  END IF;
  RETURN retcode;
END;
/
```

Log_error_Table

Purpose

This script is used for log error table.

Syntax

```
CREATE TABLE log_error (ERROR_CODE VARCHAR2(100),LOG_TIME DATE);
```

Approot_AppSeed_Creation.sql

Approot appseed sync sql command.

Prerequisites

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 Approot and ApprootSeed Configuration
SPOOL "&SPOOL_PATH"
/* Inputs are recieved */
accept P_CDB_USER Prompt 'Enter CDB Schema Username: [Eg: sys]'
accept P_CDB_PWD Prompt  'Enter CDB Schema Password: [Eg: PASSWORD]'
accept P_CDB_HOST Prompt 'Enter CDB Schema Host: [Eg: fcubs.in.oracle.com]'
accept P_CDB_PORT Prompt 'Enter CDB Schema Port: [Eg: 1521]'
accept P_CDB_NAME Prompt 'Enter CDB Schema Name: [Eg: FCUBSCDB]'
accept P_DB_MOUNTED_PATH Prompt 'Enter CDB mounted path for approot
application seed creation[Eg: /scratch/db1800dat] :'

accept P_TEMPLATE_PDB Prompt 'Enter Template PDB Name: [Eg: pdbfcubs]'
accept P_APPROOT_NAME Prompt 'Enter Approot Name: [Eg: appfcubs]'
accept P_PDB_TO_APPPDB Prompt 'Please provide path for pdb_to_apppdb.sql:
[E.g:
C:\app\client\user\product\19.0.0\client_1\rdbms\admin\pdb_to_apppdb.sql]'
accept P_COMMON_USER Prompt 'Enter Common Username created in Template PDB:
[E.g: fcubs]'

/* Connecting to cdb
conn sys/FC142SYS18C@fc142cbd as sysdba */
CONN &P_CDB_USER/&P_CDB_PWD@&P_CDB_NAME AS sysdba;

/* Creating the Approot */
CREATE pluggable database &P_APPROOT_NAME AS application container FROM
&P_TEMPLATE_PDB file_name_convert=('&P_TEMPLATE_PDB','&P_APPROOT_NAME');
ALTER pluggable database &P_APPROOT_NAME open;
```

```
/* Connecting to Approot as sysdba*/
conn &P_CDB_USER/&P_CDB_PWD@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)
(HOST=&P_CDB_HOST)(PORT=&P_CDB_PORT)))(CONNECT_DATA=(SERVER=DEDICATED)
(SERVICE_NAME=&P_APPROOT_NAME))) as sysdba;

grant select on v_$session to &P_COMMON_USER container=all;
grant create session to &P_COMMON_USER container=all;
grant select on gv_$session to &P_COMMON_USER container=all;
grant select on gv_$session to &P_COMMON_USER container=all;
grant select on v_$database to &P_COMMON_USER container=all;

/*Creating Application Seed Manually*/
create pluggable database as seed from &P_APPROOT_NAME
file_name_convert=('&P_DB_MOUNTED_PATH/&P_CDB_NAME/
&P_APPROOT_NAME/','&P_DB_MOUNTED_PATH/&P_CDB_NAME/SEED&P_APPROOT_NAME/');

alter pluggable database &P_APPROOT_NAME$SEED open;
alter session set container = &P_APPROOT_NAME$SEED;

@&P_PDB_TO_APPPDB;
select cause, type, message, status, action from pdb_plug_in_violations;

SET ERRORLOGGING OFF
SPOOL OFF
```