

# Hand-On-Lab

November 24, 2015

## Oracle Database 12c - Upgrade, Migration & Consolidation

Roy Swonger

*Senior Director and Product Manager*

*ST – Database Utilities*

*ORACLE Corporation*

Mike Dietrich

*Senior Principal Technologist*

*ST – Upgrade Development Group*

*ORACLE Corporation*

### The 4 parts of the Lab:

1. **Upgrade** an Oracle 11.2.0.4 database (SID: **UPGR**) to Oracle 12.1.0.2
2. **Plug in** the upgrade **UPGR** database into an existing Oracle 12.1.0.2 container database (SID: **CDB2**)
3. **Migrate** an Oracle 11.2.0.4 database (SID: **FTEX**) to Oracle 12.1.0.2 using Full Transportable Export/Import into a new pluggable database **PDB2**

### [optional]:

4. **Work** with Multitenant databases and implement new Oracle Database 12c features

Before you can start you may have to setup a few things and make yourself familiar with the environment

# Setup Tasks

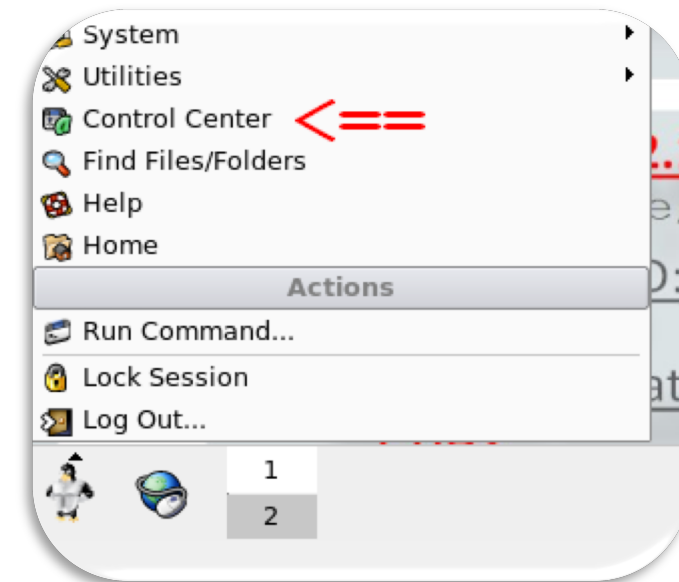
## Keyboard Layout

The default keyboard layout may be German or US English. **If you would like a different layout**, then you can add another keyboard as follows:

1. Login to the Linux desktop (user/password are both "oracle")
2. If you want to change the keyboard's layout (default is: US) to German please just **CLICK ONCE** on the tiny "US" symbol next to the clock:



3. If you'd like to do more general changes please enter the KDE CONTROL CENTER by **CLICKING** on the penguin in the left bottom corner:



## !!! IMPORTANT !!! THINGS TO KNOW AND UNDERSTAND !!!

All **passwords** are set to: **oracle**

**Switch environments** for instance: **. cdb2** (type in on shell prompt: **<dot> <blank> cdb2**)

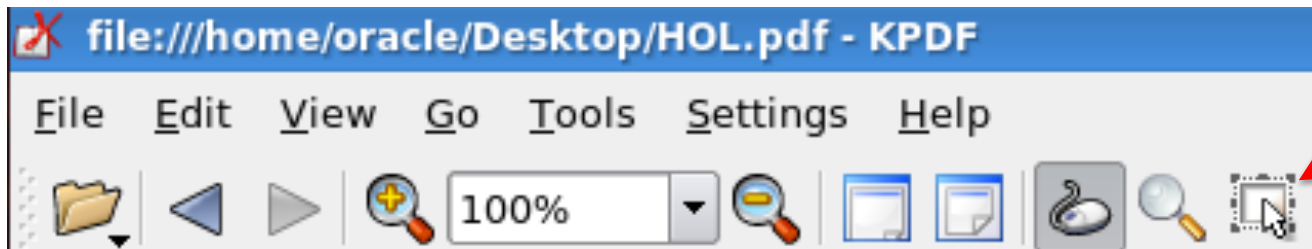
There is an environment variable **\$OH12** defined for convenience. This points to the 12.1.0.2 Oracle Home, and is used several times in part 1 of the lab.

Dark gray background, white characters mean: **Execute on the command prompt (OS shell)**

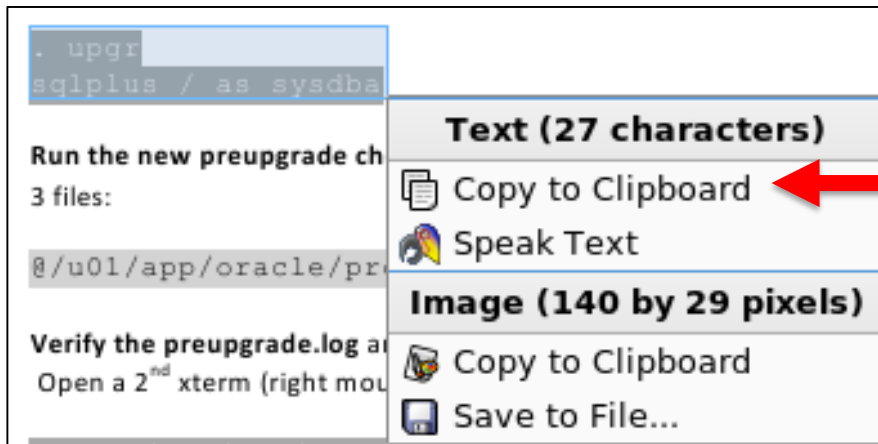
Light gray background, black characters mean: **Execute in SQL\*Plus**

## HOW TO COPY/PASTE WITH THE OKULAR PDF VIEWER

This VM includes the `okular` PDF reader, which is available in the Linux distribution. In order to copy/paste with `okular`, you need to use the “Select Tool.” If you have these instructions open within the VM, then you can see the select tool at the top of the window. It is the box with the dashed outline as shown here:



Once you choose the Select Tool, this will be the default unless you change it. Then you can select an area and copy the selection as text. Choose the Text -> “Copy to Clipboard” option as in this example from the first steps in the lab:



**IMPORTANT !!! System Overview - The numbers on the picture describe part 1-4 of the Hands-On-Lab !!!**

Oracle Database **11.2.0.4** Home:  
/u01/app/oracle/product/11.2.0

Database SID:  
**UPGR**                    \$> . upgr

Database SID:  
**FTEX**                    \$> . ftex

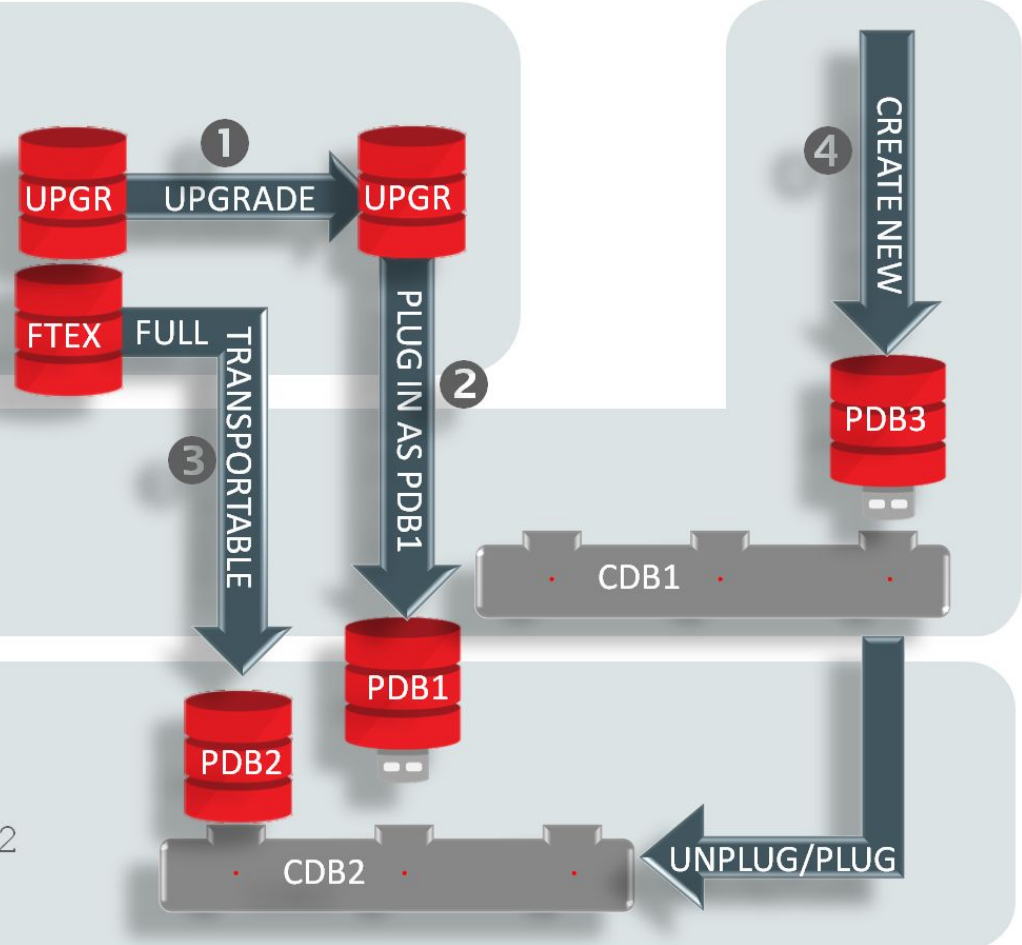
Oracle Database **12.1.0.1** Home:  
/u01/app/oracle/product/12.1.0.1

Container Database SID:  
**CDB1**                    \$> . cdb1

Oracle Database **12.1.0.2** Home:  
/u01/app/oracle/product/12.1.0.2

Database SID:  
**UPGR**                    \$> . upgr12

Container Database SID:  
**CDB2**                    \$> . cdb2



# 1

## HOL – Part 1– Upgrade the Oracle 11.2.0.4 database UPGR to Oracle Database 12.1.0.2

Database files location:	/oradata/UPGR
Initialization parameter and password file location:	/u01/app/oracle/product/11.2.0/dbs
Listener configuration:	/u01/app/oracle/product/12.1.0.2/network/admin

### Tasks HOL Part 1

Your task in HOL Part 1 will be a simple and straightforward database upgrade to Oracle Database 12c. Everything is installed already. Your Oracle Database 11g Release 2 (11.2.0.4) database (SID: **UPGR**) is startup already and ready to go. Now follow all steps and upgrade it.

You will use the new pre-upgrade check script **preupgrd.sql** which will examine your UPGR database. This script is shipped with the new Oracle 12c home in /u01/app/oracle/product/12.1.0.2/rdbms/admin

You will then prepare your UPGR database for the upgrade to Oracle Database 12c and upgrade it. The database will stay in place and doesn't get moved to another location.

#### Remarks:

For this hands-on lab we have provided easy commands to switch environments! You can switch between environments on the bash shell prompt typing ((don't type "\$>"!!!) in every Terminal/xterm:

SID: UPGR – Oracle 11.2.0.4 home	SID: UPGR – Oracle 12.1.0.2 home
<pre>\$&gt; . upgr</pre> <p>&lt;dot&gt; &lt;space&gt; upgr for the Oracle 11.2.0.4 environment with your database to be upgraded (SID: UPGR)</p>	<pre>\$&gt; . upgr12</pre> <p>&lt;dot&gt; &lt;space&gt; upgr12 for the Oracle 12c environment with your database to be upgraded (SID: UPGR)</p>

**\*\*\* START HERE \*\*\* Command Line Upgrade from Oracle 11.2.0.4 to Oracle 12.1.0.2 \*\*\***

In this section you'll execute the **new preupgrd.sql** check script, verify the output, execute some commands and a fixup script and prepare a new spfile for the upgrade. Then you'll copy the spfile and the password file to the new Oracle Database 12c home.

SID: UPGR Oracle 11.2.0.4 home	SID: UPGR Oracle 12.1.0.2 home
<p><b>Execute pre-upgrade preparation steps</b></p> <ol style="list-style-type: none"> <li>1. Open an xterm (Terminal icon) - (right mouse click → Konsole ...)</li> </ol> <pre>. upgr sqlplus / as sysdba</pre> <ol style="list-style-type: none"> <li>2. <b>Run the new preupgrade check script preupgrd.sql</b>, in your 11.2.0.4 environment – it will generate 3 files:</li> </ol> <pre>@\$OH12/rdbms/admin/preupgrd.sql</pre> <ol style="list-style-type: none"> <li>3. <b>Verify the preupgrade.log</b> and make necessary changes Open a 2<sup>nd</sup> xterm (right mouse click → Konsole ...):</li> </ol> <pre>less /u01/app/oracle/cfgtoollogs/UPGR/preupgrade/preupgrade.log</pre>	

The 11.2.0.4 database has the OLAP Catalog (AMD) component installed, and this component is no longer included in Oracle Database starting with Oracle Database 12c.

**Remove the OLAP Catalog (AMD) component** using the script from the 12.1.0.2 Oracle Home (\$OH12):

```
@$OH12/olap/admin/catnoamd.sql  
commit;
```

The preupgrade log includes a message about moving audit data from system.aud\$ to sys.aud\$ because Oracle Label Security is installed.

**Move the AUD\$ table now using the `olspreupgrade.sql`** script from the Oracle Database 12c home from SYSTEM to SYS:

```
@$OH12/rdbms/admin/olspreupgrade.sql
```

**Prepare your spfile** for the 12c upgrade according to the output from preupgrade.log:

(Please note: **Best Practice** would be to edit the `init.ora` for the upgrade manually. You could do so – the way we propose here is just a shortcut avoiding manual edit steps)

```
create pfile from spfile;  
alter system set processes=300 scope=spfile;
```

**Raise COMPATIBLE** for the upgrade, so that we can use this database later in part 2 of the lab.

```
alter system set COMPATIBLE='12.1.0' scope=spfile;
```

4. **Gather dictionary stats** prior to the upgrade:

```
EXECUTE dbms_stats.gather_dictionary_stats;
```

5. **Execute the `preupgrade_fixups.sql`** – it was created by `preupgrd.sql` in directory `/u01/app/oracle/cfgtoollogs/UPGR/preupgrade`



Be aware that the `preupgrade_fixups.sql` will display warnings about things you may have fixed already. Please ignore this as it is static information which got created during the run of `preupgrd.sql` and does not get updated when you fix issues being signaled. This is a known issue and will be changed in a future release of Oracle.

```
@/u01/app/oracle/cfgtoollogs/UPGR/preupgrade/preupgrade_fixups.sql
```

Please note that the `preupgrade_fixups.sql` script will still complain about PROCESSES being set too low. This is because we have used the ALTER SYSTEM command to adjust the parameter, but that adjustment will not take effect until the database is shutdown and restarted. Because we specified SCOPE=SPFILE, this parameter will be set correctly for the upgrade.

6. Shutdown the UPGR database:

```
shutdown immediate  
exit
```

7. Copy your new spfile and your password file into the Oracle 12c home's dbs (\$OH12/dbs) directory:

```
cp $ORACLE_HOME/dbs/spfileUPGR.ora $OH12/dbs/
```

```
cp $ORACLE_HOME/dbs/orapwUPGR $OH12/dbs/
```

SID: UPGR  
Oracle 11.2.0.4 home

SID: UPGR  
Oracle 12.1.0.2 home

### Execute all parallel upgrade steps

Now you'll upgrade your UPGR database to Oracle Database 12c using the new parallel upgrade scripts. Furthermore you'll recompile and check for invalid objects before/after the upgrade.

1. Open an xterm (Terminal icon) -  
(right mouse click → Konsole ...)

```
. upgr12  
sqlplus / as sysdba
```

2. **Bring the UPGR database into UPGRADE mode**

```
startup upgrade  
exit
```

3. **Upgrade the UPGR database** with the parallel upgrade script

Start the new parallel upgrade – **it will be driven by a PERL script catctl.pl** outside of SQL\*Plus and execute in 4 parallel threads – in maximum you could run with 8 parallel threads by specifying the parameter option **-n 8**

```
cd $ORACLE_HOME/rdbms/admin  
$ORACLE_HOME/perl/bin/perl catctl.pl catupgrd.sql
```

You will now see that as many as 73 phases will be listed – some can act in parallel, other require serial execution. **This will now take up to 15-30 minutes depending on your system.** If you wonder about the RESTART phases: those happen if timing dependencies make it necessary to rerun a certain action. The logfiles will be written by default into the directory from which you started catctl.pl, \$ORACLE\_HOME/rdbms/admin

	Once the upgrade is finished it will shutdown the database and in the next phase you'll restart it in normal mode.
	<p><b>IF YOUR MACHINE IS WELL EQUIPPED WITH RAM/CPU YOU MAY DO TASKS IN PARALLEL AND START WITH PART 3 (FULL TRANSPORTABLE EXPORT). GOTO PAGE 15 – HOL PART 3</b></p>

SID: UPGR Oracle 11.2.0.4 home	SID: UPGR Oracle 12.1.0.2 home
	<p><b>Finalize the upgrade with all required post upgrade steps</b></p> <p>During this part you'll finalize the upgrade with recompilation, postupgrade_fixups.sql and the time zone adjustment to TZ V18. Startup the database – post upgrade it is shutdown:</p> <ol style="list-style-type: none"> <li>1. Open an xterm (Terminal icon) - (right mouse click → Konsole ...)</li> </ol> <pre>. upgr12 sqlplus / as sysdba</pre> <ol style="list-style-type: none"> <li>2. <b>Startup</b> the UPGR database and recompile everything:</li> </ol> <pre>startup @?/rdbms/admin/utlrp.sql</pre>

3. **Execute the `postupgrade_fixups.sql`:**

```
@/u01/app/oracle/cfgtoollogs/UPGR/preupgrade/postupgrade_fixups.sql
```

4. **Adjust Time Zone** settings – you may look into the scripts taken from MOS Note: 1509653.1 before executing them:

```
@/home/oracle/DST/DST_prepare.sql
```

```
@/home/oracle/DST/DST_adjust.sql
```

```
exit
```

**\*\*\* COMPLETED \*\*\* Tasks HOL Part 1 \*\*\***

# 2

## HOL – Part 2 – Plug in UPGR into CDB2, an Oracle Database 12.1.0.2 container database

Database files location: /oradata/UPGR  
Initialization parameter and password file location: /u01/app/oracle/product/12.1.0.2/dbs  
Listener configuration: /u01/app/oracle/product/12.1.0.2/network/admin

### Tasks HOL Part 2

Oracle Multitenant Option is a way to **consolidate** several independent databases into one large **Container Database**. The CDB\$ROOT is the administrative layer and contains absolutely no user or application data. The PDBs that are plugged into the CDB contain the user and application data. With Oracle Database 12c Release 1 you can have up to 252 PDBs within one CDB.

Applications and clients will connect to the PDB just as they would connect to a non-CDB. The entire CDB/PDB shares one SGA, one set of background processes, one redo log stream.

In HOL Part 2 you will plug in the already upgraded UPGR database as a new pluggable database PDB1 into the **already existing Container Database CDB2**. The data files of UPGR will stay in place.

SID: UPGR – Oracle 12.1.0.2 home	SID: CDB2 – Oracle 12.1.0.2 home
<pre>\$&gt; . upgr12</pre> <p>&lt;dot&gt; &lt;space&gt; upgr12 for the Oracle 12.1.0.2 environment with your database to be plugged in later (SID: UPGR)</p>	<pre>\$&gt; . cdb2</pre> <p>&lt;dot&gt; &lt;space&gt; cdb2 for the Oracle 12c environment connecting to the Container Database (SID: CDB2)</p>

\*\*\* START HERE \*\*\* Plug in UPGR into CDB2 \*\*\*

In this section an XML description file for UPGR will be created and used to plug UPGR into CDB2 as new PDB1. Finally sanity operations will have to be done to assimilate UPGR finally as PDB2.

Please note: There's no ALTER PLUGGABLE DATABASE ... RECONVERT command available. To migrate a database back into a stand-alone database either Data Pump, Transportable Tablespaces or similar techniques will need to be used.

SID: UPGR Oracle 12.1.0.2 home	SID: CDB2 Oracle 12.1.0.2 home
<p><b>Prepare the UPGR database for plug in</b></p> <ol style="list-style-type: none"><li>1. Switch to the UPGR Oracle 12.1.0.2 environment: <pre>. upgr12 sqlplus / as sysdba</pre></li><li>2. <b>Start</b> the UPGR database in <b>read-only</b> mode: <pre>shutdown immediate startup open read only;</pre></li><li>3. Generate the XML description file – this file will contain the information describing the database structure. To create it the database UPGR has to be in read only mode: <pre>exec DBMS_PDB.DESCRIBE ('/tmp/pdb1.xml');</pre></li><li>4. Shutdown the database <pre>shutdown immediate exit</pre></li></ol>	

SID: UGR  
Oracle 12.1.0.2 home

SID: CDB2  
Oracle 12.1.0.2 home

#### Prepare the UGR database for plug in

1. Switch to the CDB2 Oracle 12.1.0.2 environment:

```
. cdb2  
sqlplus / as sysdba
```

2. Startup the CDB2, a precreated Oracle 12c (12.1.0.2) Container Database:

```
startup
```

3. Check plug in compatibility first:

```
SET SERVEROUTPUT ON
```

```
DECLARE  
compatible CONSTANT VARCHAR2(3) := CASE  
DBMS_PDB.CHECK_PLUG_COMPATIBILITY( pdb_descr_file => '/tmp/pdb1.xml',  
pdb_name => 'PDB1') WHEN TRUE THEN 'YES' ELSE 'NO'  
END;  
BEGIN  
DBMS_OUTPUT.PUT_LINE(compatible);  
END;  
/
```

4. Now plug in the database with its new name **PDB1** – from this point there's no UGR database anymore. In a real world environment you would have a backup or use a backup/copy to plug in. In our lab the database UGR will stay in place and become PDB1 as part of CDB2. Please use the proposed naming as the TNS setup has been done already. Use the NOCOPY option for this lab to avoid additional 2-3 minutes copy time

```
create pluggable database PDB1 using '/tmp/pdb1.xml' nocopy tempfile
reuse;
```

5. Connect to this new PDB1 and perform sanity operations:

```
alter session set container=PDB1;
@?/rdbms/admin/noncdb_to_pdb.sql
```

Sanity operations required inside the PDB to connect the PDB with the CDB correctly. Therefore run the script `noncdb_to_pdb.sql` – this may take a approx **10-20 minutes** to complete due to recompilations. If the script didn't get executed the PDB1 would open in restricted mode only.

5. Now the database UPGR is plugged in – but not open yet. It will need to be started.

```
startup
show pdbs
exit
```

6. To connect to the consolidated PDB1 from the command prompt the following command syntax needs to be used:

```
sqlplus "sys/oracle@pdb1 as sysdba"
```

*As an **alternative** you could use the **EZconnect** syntax:*

```
sqlplus "sys/oracle@//localhost:1521/pdb1 as sysdba"
```

\*\*\* COMPLETED \*\*\* Tasks HOL Part 2 \*\*\*



# 3

## HOL – Part 3 – Migrate FTEX database with Full Transportable Export/Import into PDB2

Database files location:	/oradata/FTEX
Pluggable database files location:	/oradata/CDB2/pdb2
Initialization parameter and password file location:	/u01/app/oracle/product/11.2.0.4/dbs
Listener configuration:	/u01/app/oracle/product/12.1.0.2/network/admin

### Tasks HOL Part 3

Full Transportable Export/Import is a new Oracle Database 12c upgrade and migration feature combining the speed of Transportable Tablespaces with the ease-of-use of Data Pump taking care of all metadata and non-transportable data.

Your task in HOL Part 3 will be to use Full Transportable Export/Import to migrate the existing Oracle 11.2.0.4 database FTEX into a new PDB2 which will belong to the container database CDB2. Please stay with the proposed names (PDB2) as the TNS setup has been set up already to allow connections etc.

This feature works independent of Oracle Multitenant and platform and can be used to migrate cross Endianness as well. Source database version has to be at least Oracle 11.2.0.3, target version needs to be at least Oracle 12.1.0.1. For cross-platform migrations RMAN backups with CONVERT operations will be necessary.

SID: FTEX – Oracle 11.2.0.4 home	SID: CDB2 – Oracle 12.1.0.2 home
<pre>\$&gt; . ftex</pre> <p>&lt;dot&gt; &lt;space&gt; ftex for the Oracle 11.2.0.4 environment with your database to be plugged in later (SID: FTEX)</p>	<pre>\$&gt; . cdb2</pre> <p>&lt;dot&gt; &lt;space&gt; cdb2 for the Oracle 12c environment connecting to the Container Database (SID: CDB2)</p>

**\*\*\* START HERE \*\*\* Migrate FTEX with Full Transportable Export/Import into PDB2 \*\*\***

The first task in the lab will be to provide an empty database – something we would do for a full import or for transportable tablespaces as well. But in this specific case we want to consolidate, and therefore pre-create an empty PDB2 (a Pluggable Database) inside the already existing CDB2 (the Container Database).

SID: FTEX Oracle 11.2.0.4 home	SID: CDB2 Oracle 12.1.0.2 home
	<p><b>Provision PDB2 from PDB\$SEED:</b></p> <ol style="list-style-type: none"><li>1. Switch to the Oracle 12c <b>CDB2 environment:</b></li></ol> <pre data-bbox="678 675 1043 743">. cdb2 sqlplus / as sysdba</pre> <pre data-bbox="678 786 1977 815">startup [ONLY ISSUE THIS COMMAND IF YOU ARE STILL RUNNING HOL PART1]</pre> <ol style="list-style-type: none"><li>2. <b>Create a new pluggable database PDB2:</b></li></ol> <p>The easiest way to create an empty PDB is to clone it from the template PDB called <b>PDB\$SEED</b> which exists in every container database. The location to create it is defined by the init parameter <b>PDB_FILE_NAME_CONVERT</b>. It is already set in CDB2 to have the new and empty PDB2 created in <b>/oradata/CDB2/pdb2</b> directory.</p> <p>Create an empty PDB by cloning the PDB\$SEED:</p> <pre data-bbox="678 1214 1939 1283">create pluggable database PDB2 admin user adm identified by adm file_name_convert=('/oradata/CDB2/pdbseed', '/oradata/CDB2/pdb2');</pre> <p>This will take 1-2 minutes.</p>

	<p>Start the new pluggable database PDB2:</p> <pre>alter session set container=pdb2; startup</pre> <p>3. Create a directory object and a database link inside the PDB2 – you will need this for the full transport operation - the directory <code>/oradata/CDB2/mydir</code> has been precreated as well for Data Pump</p> <pre>create directory mydir as '/oradata/CDB2/mydir'; grant read, write on directory mydir to system; create public database link SOURCEDB connect to system identified by oracle using 'FTEX'; exit</pre>
--	--

SID: FTEX Oracle 11.2.0.4 home	SID: CDB2 Oracle 12.1.0.2 home
<p><b>Prepare the FTEX database for the Full Transportable Export/Import</b></p> <p>In order to run the Full Transportable operation we must set all data tablespaces into read-only mode. This is the same procedure we would follow for a regular transportable tablespace operation. Once the tablespace is in read-only mode we can copy the file(s) to the target location</p> <p>1. Switch to the Oracle 12c UPGR environment:</p> <pre>. ftex sqlplus / as sysdba</pre>	

2. **Start the FTEX database and switch data tablespaces (here: USERS) into read-only mode:**

```
startup  
alter tablespace users read only;  
exit
```

3. Copy the files to the target location

```
cp /oradata/FTEX/users01.dbf /oradata/CDB2/pdb2
```

SID: FTEX  
Oracle 11.2.0.4 home

SID: CDB2  
Oracle 12.1.0.2 home

#### Data migration via Full Transportable Export/Import from FTEX into PDB2

The Data Pump import will be run through the database link you created earlier – thus no need for an export or a dumpfile. Data Pump will take care of everything (currently except XDB and AWR) you need from the system tablespaces and move views, synonyms, trigger etc over to the target database (in our case: PDB2).

1. Switch to the Oracle 12c **CDB2 environment:**

```
. cdb2
```

2. **Execute the Full Transportable Export/Import with Data Pump**

```
impdp system/oracle@pdb2 network_link=sourcedb version=12 full=y \  
transportable=always metrics=y exclude=statistics directory=mydir \  
logfile=pdb2.log \  
transport datafiles='/oradata/CDB2/pdb2/users01.dbf'
```

In case copy&paste does not work we have prepared a par file in /home/oracle/IMP.  
The PDB2 is open and ready to use after the Transport migration has completed:

```
sqlplus "system/oracle@PDB2"
```

\*\*\* COMPLETED \*\*\* Tasks HOL Part 3 \*\*\*

## 4 [optional]

### HOL – Part 4 – Create PDB3 in Oracle 12.1.0.1 and upgrade via plug out/in to Oracle 12.1.0.2

Database files location:	/oradata/CDB1/pdb3
Pluggable database files location:	/oradata/CDB2/pdb3
Initialization parameter and password file location:	/u01/app/oracle/product/12.1.0.1/dbs
Listener configuration:	/u01/app/oracle/product/12.1.0.2/network/admin

#### Tasks HOL Part 4

One technique to upgrade pluggable databases in a Multitenant environment is unplug-plugin. This approach gives a lot of control over a pluggable database upgrade but requires manual steps, similar to the command line upgrade.

In this part of the HOL a new PDB will be created in an Oracle 12.1.0.1 CDB1 and upgraded via unplug/plugin into the Oracle 12.1.0.2 CDB2.

SID: CDB1 – Oracle 12.1.0.1 home	SID: CDB2 – Oracle 12.1.0.2 home
<pre>\$&gt; . cdb1</pre> <p>&lt;dot&gt; &lt;space&gt; cdb1 for the Oracle 12.1.0.1 environment with your database to be plugged in later (SID: CDB1)</p>	<pre>\$&gt; . cdb2</pre> <p>&lt;dot&gt; &lt;space&gt; cdb2 for the Oracle 12c environment connecting to the Container Database (SID: CDB2)</p>

\*\*\* START HERE \*\*\* Create PDB3, upgrade it to Oracle 12.1.0.2 via plug out/in \*\*\*

The first task in the lab will be to provide an empty database – something we would do for a full import or for transportable tablespaces as well. But in this specific case we want to consolidate, and therefore pre-create an empty PDB2 (a Pluggable Database) inside the already existing CDB2 (the Container Database).

SID: CDB1 → PDB3 Oracle 12.1.0.1 home	SID: CDB2 → PDB3 Oracle 12.1.0.2 home
<p><b>Create a new pluggable database PDB3:</b></p> <ol style="list-style-type: none"><li>1. Switch to the Oracle 12.1.0.1 <b>CDB1 environment:</b></li></ol> <pre data-bbox="241 675 611 743">. cdb1 sqlplus / as sysdba</pre> <ol style="list-style-type: none"><li>2. <b>Start the CDB1 container database</b> – it has no PDBs yet (except for PDB\$SEED):</li></ol> <pre data-bbox="241 842 383 874">startup</pre> <ol style="list-style-type: none"><li>3. <b>Create a new pluggable database PDB3 and start it:</b></li></ol> <pre data-bbox="241 999 1503 1070">create pluggable database PDB3 admin user adm identified by adm file_name_convert=('/oradata/CDB1/pdbseed', '/oradata/CDB1/pdb3');</pre> <p data-bbox="241 1078 584 1110">This will take 1-2 minutes.</p> <pre data-bbox="241 1153 875 1222">alter session set container=pdb3; startup</pre> <ol style="list-style-type: none"><li>4. Execute the preupgrade check script</li></ol> <pre data-bbox="241 1342 1357 1374">@/u01/app/oracle/product/12.1.0.2/rdbms/admin/preupgrd.sql</pre>	

4. **Verify the preupgrade.log** and make necessary changes  
Open a 2<sup>nd</sup> xterm (right mouse click → Konsole ...):

```
less /u01/app/oracle/cfgtoollogs/CDB1/preupgrade/preupgrade.log
```

5. **Gather dictionary stats** prior to the upgrade:

```
EXECUTE dbms_stats.gather_dictionary_stats;
```

6. **Execute the preupgrade\_fixups.sql** – it got created by preupgrd.sql run in directory /u01/app/oracle/cfgtoollogs/CDB1/preupgrade

```
@/u01/app/oracle/cfgtoollogs/CDB1/preupgrade/preupgrade_fixups.sql
```

7. Switch to the CDB\$ROOT layer, **close** the pluggable database PDB3 and **unplug** it

```
alter session set container=CDB$ROOT;  
alter pluggable database PDB3 close;  
alter pluggable database PDB3 unplug into '/tmp/pdb3.xml';  
drop pluggable database PDB3 keep datafiles;
```

```
shutdown immediate  
exit
```

SID: CDB1 → PDB3 Oracle 12.1.0.1 home	SID: CDB2 → PDB3 Oracle 12.1.0.2 home
	<b>Plug in the PDB3 into CDB2 and upgrade it to Oracle 12.1.0.2:</b>  1. Switch to the Oracle 12.1.0.2 CDB2 environment:  <pre>. cdb2</pre>



```
sqlplus / as sysdba
```

2. Execute the **Plug In Check** and check **PDB\_PLUG\_IN\_VIOLATIONS** when the result of the plug in check is **"NO"**:

```
SET SERVEROUTPUT ON
```

```
DECLARE
  compatible CONSTANT VARCHAR2(3) := CASE
DBMS_PDB.CHECK_PLUG_COMPATIBILITY(
  pdb_descr_file => '/tmp/pdb3.xml',
  pdb_name => 'PDB3')
  WHEN TRUE THEN 'YES' ELSE 'NO'
END;
BEGIN
DBMS_OUTPUT.PUT_LINE(compatible);
END;
/
```

```
select message, status from pdb_plug_in_violations where type like
'%ERR%';
```

3. Plug in the PDB3 into CDB2

```
create pluggable database pdb3 using '/tmp/pdb3.xml' file_name_convert=(
'/oradata/CDB1/pdb3', '/oradata/CDB2/pdb3');
```

4. Open PDB3 in UPGRADE mode and upgrade it

```
alter pluggable database PDB3 open upgrade;
exit
```

```
cd $ORACLE_HOME/rdbms/admin
$ORACLE_HOME/perl/bin/perl catctl.pl -c 'PDB3' catupgrd.sql
```

#### 5. Recompile after upgrade

```
sqlplus / as sysdba

alter session set container=PDB3;
startup
@?/rdbms/admin/utlrp.sql
show pdbs

exit
```

NN SID: CDB2 → PDB3  
Oracle 12.1.0.2 home

#### Finally a few CDB/PDB exercises

First test will introduce you to the new CDB views. Therefore we create a simple table and check its visibility within the dictionary views

1. Connect directly to **PDB1** in the **Oracle 12.1.0.2** environment:

```
. cdb2
sqlplus "sys/oracle@//localhost:1521/pdb1 as sysdba"
```

2. Create a table and insert data:

```
create table HOL (col1 number);
insert into HOL values (1);
commit;
```

3. Connect directly to **PDB2**:

```
alter session set container=PDB2;
```

4. Create a table and insert data:

```
create table HOL (col1 number);  
insert into HOL values (2);  
commit;
```

5. Connect directly to **PDB3**:

```
alter session set container=PDB3;
```

6. Create a table and insert data:

```
create table HOL (col1 number);  
insert into HOL values (2);  
commit;
```

7. Connect directly to **CDB\$ROOT**:

```
alter session set container=cdb$root;
```

8. Query the data from the CDB\_VIEWS:

```
select CON_ID, SUBSTR(TABLE_NAME,1,10) TNAME from CDB_TABLES WHERE TABLE_NAME='HOL';
```

You'll see that each table HOL within a certain PDB is visible to the CDB\$ROOT. But if you'd repeat the exercise within each of the PDBs you'll see just the contents on a PDB level. Recognize the CON\_ID which represents where an object exists.

\*\*\* COMPLETED \*\*\* Tasks HOL Part 4 \*\*\*

Thank you for completing our Upgrade, Migrate & Consolidate to Oracle Database 12c Hands-On-Lab.

If you have further questions you may please download the +500 slide deck containing almost everything about upgrades and migrations. And always feel free to contact us directly.

<http://blogs.oracle.com/UPGRADE>

Thanks and successful upgrades!

Roy Swonger & Mike Dietrich & The Database Upgrade Team