# Move Pdb on SOP

# Introduction

The Standard Oracle Platform is setup using Oracle MAA (Maximum Availability Architecture) featuring Dataguard and Real Application Cluster.

All databases are setup in a Mulitenant environment where one Container can contain many pluggable databases (pdb’s). During normal operation issues can occur that will require that a Pdb will move from one Container toanother Container.

For example the following can demand a move from one container to another.’:

* Issues on the container or cluster and we would like to evacuate the container.
* A Pdb demand that the system parameter for the Container is modified.
* A Pdb needs patching
* A Pdb needs to be upgraded to a new relase.

This document describes one way of moving a Pdb to a new container.

# Preparing the Move

The software is a bunch of scripts running in sequel from both the source and the primary database. In case of a standby (typically pre production and production environments) additional scripts must be run on the standby nodes as well.

Scripts are numbered 1\_..., 2\_.... etc and must be executed in numerical order.

Software can be fetched from git but there is a copy on db-s007hh01d.

Simply use scp -r [oracle@db-s007hh01d.qaoneadr.local:/clustershare/move\_pdb](mailto:oracle@db-s007hh01d.qaoneadr.local:/clustershare/move_pdb) .

The catalog structure looks like:  
  
move\_pdb (root, contains only catalogs)

* conf (Contains one configuration file for the scripts)
* source\_primary (Scripts to run from source primary)
* source\_standby (Scripts to run from source standby)
* target\_primary (Scripts to run from target primary)
* target\_standby (Scripts to run from target standby)
* Various helper scripts

Every script will generate a log file at least telling when the script has been started and when it has been stopped.

For convenience copy the move\_pdb script to the current database host and the to-be database host. Also copy the scripts to standby hosts if standby is in use.

## The Conf file

The Configuration catalog (conf) only contains one configuration file:  
The file is named move\_pdb.conf.

The file contains the following:

|  |  |
| --- | --- |
| Parameter | Explanation |
| CDB\_SOURCE | Container name for the source |
| CDB\_TARGET | Container name for target |
| CDB\_SOURCE\_STANDBY | Container name for the source standby |
| CDB\_TARGET\_STANDBY | Container name for the target standby |
| CDB\_RUNNING\_INSTANCES | Instances running the database on the source |
| CDB\_MIGRATE\_SERVICE | A running service on the source container |
| CDB\_REMOTE\_USER | A C## User on the Source container |
| CLONE\_LINK | Name of Database Link |
| VIP | VIP address used by database link connecting the target to the source CDB |
| PDB\_SOURCE | Name of the PDB on the Source CDB |
| PDB\_TARGET | Name of the PDB on the Target CDB (can be the same as the PDB\_SOURCE) |
| PDB\_PREFERRED\_INSTANCE | Name of of the (new) preferred instance on target. Only used in case of RAC target |
| PDB\_AVAILABLE\_INSTANCE | Name of of the (new) available instance on target. Only used in case of RAC target |
| PDB\_MIGRATE\_SERVICE | Name of temporary migration service. Must be unique on source cluster |
| PDB\_PRIMARY\_SERVICE | Name of Primary service |
| PDB\_STANDBY\_SERVICE | Name of Read only service |
| PDB\_NODBOT\_SERVICE | Name of service used by Nodbot |

Every field must be entered in respect to the source and the target databases.

The scripts can be devided into three subsections:

* Prepare the Source
* Move the Pdb to the Target
* Handle the (optional) standby database.
* Clean up

## Prepare the Source

The source needs to be prepared before the migrations. Also we do have some prerequisite to check for.

### Step 1: Precheck

It is important to check the status for the source before the move to the new container.

The script **1\_pre\_check\_source.sh** checks for:

* Materialized view connected to remote databases

If there is a fast refreshable link the CI team needs to ensure that the materialized view log on the target database is maintained. Usually the source needs to be deregistered.

* Invalid objects

Invalid objects on SYS needs to be handled. User objects with errors can be ignored.

* Local Undo

Database must have local undo enabled.

* Log Mode

The database has to be running archive log mode.

Aside from that the scripts report the size of the database and various system parameters.

The DBA needs to validate the log file and ensure that the prerequisites are in place.

### Step 2: Create a user

The user needs to be created on the container level and will be used by a database link created on the target container database. The user will have very strong permissions and will be dropped during the cleanup phase.

The script **2\_create\_user\_clone.sh** is used to create the user.

### Step 3: Create a temporary migration service for the Source PDB

If a PDB does not have a running service there is a possibility that the PDB will be shutdown during the migration. To prevent this from happening we create a PDB migration service to prevent this from happening. The service will be removed during the cleanup phase.

The script **3\_create\_migrate\_service.sh** is used to create the service.

### Step 4: Open the Source PDB in Read Only mode

During the migration we need to ensure that the source database is open and not being updated. Naturally this restriction will be lifted when we move to strictly 19 databases.

The script **4\_open\_read\_only.sh** is used to reopen the database in read only mode.

### Step 5: Start the Migration Service

After the database has been shutdown we need to ensure that the migration service are running.

The script **5\_start\_migrate\_service.sh** starts the migrating services.

During these 5 steps the source has been prepared and we are moving on to the new target database.

## Move The Pdb to the Target Database

Step 6: Create the Database Link

The database link will be used to create the PDB on the Target. The database link will connect to the (read only) source database.

The script **6\_create\_db\_link.sh** creates the database link.

### Step 7: Create The Target PDB

Simply clone the Source PDB to the Target PDB using the database link.

The script **7\_create\_pdb.sh** creates the database on the target. The script can take a long time and I do think this should be run using nohup.

### Step 8: Validate the Target after the PDB has been created

This step simply validates if the database is created. The script should return the new PDB in various instances. The open mode are expected to be MOUNT.

The script **8\_pdb\_info.sh**

## Patch or Upgrade

So now we do have the Target PDB populated and need to determine if we need to patch or upgrade the Target PDB. If we have moved from one version to another (12.2 => 19.9) we need to upgrade, if we have moved from 12.2 to another 12.2 home we (probably) need to patch the PDB.

### Patch

It is not guaranteed that we need to patch the database.

Opening the database will inform us. Also we need to verify if we have moved from a RAC cluster to a non RAC cluster. In that case we do need to turn the RAC feature off on the PDB.

#### Step 9: Open the PDB (for patching)

The PDB will open the database and will return errors if patching is needed.  
The DBA needs to verify the log for the script.

The script **9\_prepare\_patch.sh** opens the database.

#### Step 10: Verify the Target PDB

Verify will check for RAC options and for required Sync between the PDB and CDB. Also it will inform if we need to patch. If the script returns no rows the patch is not necessary.

The script **10\_check\_pdb\_violations.sh** verifies the database.

#### Step 11: Execute the datapatch

If Step 10 reports issues we need to patch the database.

This can be done using the script **11\_execute\_datapatch.sh**

It is very important that the DBA validates the logfile for this script.

### Upgrade

If we change the database version we need to upgrade the database.

This is done using standard Oracle tools.

#### Step 12: Open the PDB for upgrade

The PDB will be opened for upgrade.

The script **12\_prepare\_upgrade.sh** will open the database.

#### Upgrade the database

Upgrade is standard Oracle:

The target environment are already set so:

cd $ORACLE\_HOME/rdbms/admin

$ORACLE\_HOME/perl/bin/perl catctl.pl -c 'PDB\_TARGET' catupgrd.sql

PDB\_TARGET can be retrieved from the configuration file move\_pdb.conf

## Post operations on the Target Primary database

Post operations

### Shutdown the database

The Pluggable Database should be opened using attached service. To ensure that is working as expected, the PDB are shutdown.

The script **13\_post\_pdb.sh** are used to shutdown the database.

### Add the service

The script **14\_add\_service.sh** creates the services and start the services.

# Handling the Standby database

If the target database is protected using a standby database (typically in production or pre-production) we do need to add additional steps on the standby side of the target database.

The nddbctl simplifies the steps necessary to complete the dataguard:

The script **15\_dgfinish.sh** can be used to complete the dataguard setup. The script can take some time and again I strongly suggest that You use nohup for this task.

# Clean Up the Source

After the migration it is important to release the resources used by the old database. Also it is important to ensure that the database is not in use and won’t be started up by accident.

The script **17\_cleanup\_service.sh** is used to stop and disable the services for the old primary database.

The script **18\_cleanup\_standby\_service.sh** is used to stop and disable the services for the old primary database.

The script 19\_remove\_source.sh will be used to destroy the source database.

The script 20\_remove\_standby.sh will be used to destroy the source standby

# Update the DNS entries

Clients connecting to the database connect to a scan listener. Every database has a TNS connection using DNS alias for the scan listeners. If the database is moved from one cluster to another it is important to update the DNS alias.

The DNS alias are named ora-<db>-a.[qaoneadr.local|oneadr.net] and   
ora-<db>-a.[qaoneadr.local|oneadr.net]. ITSSP can be used to update the DNS alias:  
  
Use the following URL: **http://ioweb.oneadr.net/itssp**

In the search bar search for: **DNS Services - Modify CNAME**

Enter the DNS CNAME in the Record field.

Under Modification choose Target – New

A new field Target – New is displayed.   
Enter the new scan listener for the cluster (eg. db-s009hhd-scan.qaoneadr.local)

Under Business Justification enter a description (move of PDB to new cluster due to … and enter the Change ID).

The design is that ora-<db>-a => scan listener in site HH  
and that ora-<db>-b => scan listener in site OE

APPENDIX A:  
  
An Example of a move\_pdb.conf configuration file:

**CDB\_SOURCE=s0281d1o**

**CDB\_TARGET=h0108d1h**

**CDB\_SOURCE\_STANDBY=s0281d1h**

**CDB\_TARGET\_STANDBY=h0108d1o**

**CDB\_RUNNING\_INSTANCES=s0281d1o**

**CDB\_MIGRATE\_SERVICE=s0281d**

**CDB\_REMOTE\_USER=c##cloneuc4d**

**CLONE\_LINK=clone\_uc4d\_link**

**VIP=db-s008oe02d-vip.qaoneadr.local**

**PDB\_SOURCE=P\_UC4\_UC40\_20091614\_D**

**PDB\_TARGET=P\_UC4\_UC40\_20091614\_D**

**PDB\_PREFERRED\_INSTANCES=h0108d1h1**

**PDB\_AVAILABLE\_INSTANCES=h0108d1h4**

**PDB\_MIGRATE\_SERVICE=pdb\_uc4d\_migrate**

**PDB\_PRIMARY\_SERVICE=uc4d**

**PDB\_STANDBY\_SERVICE=uc4d\_ro**

**PDB\_NODBOT\_SERVICE=s\_uc4\_uc40\_20091614\_d**