DELPHIX

# Provisioning vPDBs from a non-MT dSource

July 2020

## Overview

Oracle 20c will only support the multitenant architecture. As a result, Oracle database customers must begin developing and testing PDB conversions from non-multitenant production sources. Delphix customers would like to use Delphix virtualized databases to test the upgrade and conversion by using non-multitenant sources and creating virtual Oracle multitenant PDBs. This document describes the steps to support this workflow.

## Pre-requisites

1. 6.0.3.0 Delphix Engine with a new feature flag `ORACLEMTCONVERT` enabled.

## Environment Requirements

1. Source host with a non-multitenant Oracle 11g or newer source database.
2. VDB target host for provisioning a virtual non-multitenant VDB from the source database.
3. CDB target host with a running Oracle target version CDB.

If testing a non-multitenant to multitenant conversion where an upgrade is also required (e.g. source is 12.2 and target is 19c) then there are two options for upgrading the database:

Upgrade Option 1: After provisioning the VDB.
Upgrade Option 2: After plugging into the CDB target database.

Option 1 requires the ability to upgrade to the Oracle target version on the VDB target host.

## How to Enable

There are three key steps to set up to enable support for this feature:

1. **Feature Flag:** A new feature flag `ORACLEMTCONVERT` will be introduced to enable this feature. This feature will be disabled by default, and needs to be enabled via the CLI. Once the feature flag `ORACLEMTCONVERT` is enabled on the Delphix Engine, it will remain enabled until it is manually disabled.
2. **Pre-snapshot Hook on VDB:** This hook will open the database in "read only" mode and issue the `DBMS_PDB.DESCRIBE` procedure to prepare the virtual database to be converted to a PDB.
3. **Non-CDB to PDB Script:** This script will run as a hook present on the target CDB host. This should call into the Oracle script `$ORACLE_HOME/rdbms/admin/noncdb_to_pdb.sql` to convert the VDB into a PDB.

## Workflow Steps

These are the steps to follow to use this functionality. These steps may be performed via the Delphix API.

1. Link the non-multitenant Oracle 11g or newer source database as a dSource within Delphix.
2. Provision a non-multitenant Oracle VDB from the dSource onto the VDB target host. This will be referred to as the **Golden VDB**.
3. (If using Upgrade Option 1) Upgrade the **Golden VDB** to the Oracle target version: manually upgrade the database and point it to the new Oracle home. This step is only necessary if the source and target Oracle versions are not the same and the data files will not be upgraded when they are converted below.
4. Create a Golden VDB Pre-snapshot hook on the **Golden VDB** to open the database in "read only" mode and issue the `DBMS_PDB.DESCRIBE` procedure to prepare the virtual database to be migrated into a PDB and create an XML file called `delphix_plugin.xml` in the VDB datafile directory.
5. (Optional) Create a Golden VDB Post-snapshot hook to remove the database from read only mode. The **Golden VDB** can also remain "read only".
6. Take a snapshot of the **Golden VDB.**
7. Create a PDB conversion script named `dx-post-plug-hook.sh` in the root of the Delphix toolkit directory of the target CDB host. The name of the PDB being provisioned/converted will be supplied by Delphix as the first parameter to the script when it invokes the script. The VDB datafiles will have already been plugged into the target CDB at the time the script is invoked. The script should do the following:
    a. (If using Upgrade Option 2) Upgrade the PDB datafiles prior to the conversion.
    b. Call into $`ORACLE_HOME/rdbms/admin/noncdb_to_pdb.sql` and perform any customizations for the multitenant conversion.
8. Select a **snapshot** (point-in-time not supported) on the **Golden VDB** that has the `delphix_plugin.xml` file and provision a virtual PDB to a Physical CDB on the CDB target host by selecting the PDB Provision Parameters as `OracleMultitenantProvisionParameters`.
   Note: This step can be executed via the **API / CLI only**, and will not be allowed via the Delphix UI.
9. To refresh the data in the virtual PDB from production, first refresh the **Golden VDB** from the dSource, then refresh the virtual PDB from the new snapshot in the **Golden VDB**.

# Sample Customer Scripts

## Golden VDB Pre-Snapshot Hook

```
#!/bin/sh


sqlplus "/ AS SYSDBA" <<-EOF

  whenever sqlerror exit 2;

  spool
$DELPHIX_MOUNT_PATH/$DELPHIX_DATABASE_UNIQUE_NAME/datafile/presnapshot
.out replace

  shutdown immediate

  startup mount

  alter database open read only;

  exec
dbms_pdb.describe(pdb_descr_file=>'$DELPHIX_MOUNT_PATH/$DELPHIX_DATABA
SE_UNIQUE_NAME/datafile/delphix_plugin.xml');

  exit;

EOF

exit_status=$?

if [ $exit_status -eq 0 ]; then

  exit 0

else

  exit $exit_status

fi
```

## Golden VDB Post-Snapshot Hook (Optional)

```
#!/bin/sh


sqlplus "/ AS SYSDBA" <<-EOF

  whenever sqlerror exit 2;
  spool
$DELPHIX_MOUNT_PATH/$DELPHIX_DATABASE_UNIQUE_NAME/datafile/postsnapsho
t.out replace

  shutdown immediate

  startup

  exit;

EOF

exit_status=$?

if [ $exit_status -eq 0 ]; then

  exit 0

else

  exit $exit_status

fi
```

## PDB Conversion Script

```
#!/bin/sh

DELPHIX_PDB_NAME=$1

SCRIPT_DIR="$( cd "$( dirname "$0" )" && pwd )"

sqlplus "/ AS SYSDBA" <<-EOF

  whenever sqlerror exit 2;

  spool $SCRIPT_DIR/$DELPHIX_PDB_NAME-pdbconvert.out replace

  alter session set container=$DELPHIX_PDB_NAME;

  @$ORACLE_HOME/rdbms/admin/noncdb_to_pdb.sql

  exit;

EOF


exit_status=$?


if [ $exit_status -eq 0 ]; then
```

```
    exit 0
else
    exit $exit_status
fi
```

## PDB Provision Parameters

(Note: values in bold below indicate values specified by the user)

```
delphixengine database provision *> ls
Properties
    type: OracleMultitenantProvisionParameters
    container:
        type: OracleDatabaseContainer
        name: <Target-VPDB-Name>
        description: (unset)
        diagnoseNoLoggingFaults: true
        group: <Target-Group>
        performanceMode: DISABLED
        preProvisioningEnabled: false
        sourcingPolicy: (unset)
    credential: (unset)
    masked: (unset)
    maskingJob: (unset)
    source:
        type: OracleVirtualPdbSource
        name: (unset)
        allowAutoVDBRestartOnHostReboot: <true || false>
        config: (unset)
        customEnvVars: (unset)
        fileMappingRules: (unset)
        logCollectionEnabled: false
        mountBase: <VPDB-Mount-Base>
        operations: (unset)
    sourceConfig:
        type: OraclePDBConfig
        cdbConfig: <Target-CDB-SourceConfig>
        databaseName: <Target-VPDB-Name>
        environmentUser: <Target-CDB-Environment-User>
        linkingEnabled: true
        nonSysCredentials: (unset)
        nonSysUser: (unset)
        Repository: <Target-CDB-Repository>
        services: (unset)
    timeflowPointParameters:
        type: TimeflowPointSemantic
        container: <Golden-VDB>
        location: LATEST_SNAPSHOT
    username: (unset)
    virtualCdb: (unset)
```

Sample CLI input file:

```
database/provision
set container.name=<Target-VPDB-Name>
set container.group=<Target-Group>
set source.allowAutoVDBRestartOnHostReboot=<true || false>
set source.mountBase=<VPDB-Mount-Base>
set sourceConfig.databaseName=<Target-VPDB-Name>
set sourceConfig.cdbConfig=<Target-CDB-SourceConfig>
set sourceConfig.environmentUser=<Target-CDB-Environment-User>
set sourceConfig.repository=<Target-CDB-Repository>
set timeflowPointParameters.type=TimeflowPointSemantic
set timeflowPointParameters.container=<Golden-VDB>
set timeflowPointParameters.location=LATEST_SNAPSHOT
commit
exit
```

Note: The `timeflowPointParameters type` can also be `TimeflowPointSnapshot` to point to a specific snapshot. Picking an arbitrary point-in-time between snapshots is not supported.

## Notes and Restrictions

This feature is not available via the Delphix UI.

The provision timeflow point must correspond to a consistent snapshot and not an arbitrary point between snapshots. The snapshot must also include a `delphix_plugin.xml` file corresponding to the files in the snapshot generated using `dbms_pdb.describe`. A pre-snapshot hook is a convenient way to accomplish this but it can also be done manually.

The target CDB must be a physical CDB. Virtual CDB targets are not supported.

Delphix does not enforce a time-out when invoking the PDB conversion script.

Delphix enforces a "<=" relationship between the source and target Oracle database versions. This is to allow the conversion script to optionally perform an upgrade.

## RAC

There are a some workflow customizations required for RAC databases:

1. The PDB conversion script must be in the root of the Delphix toolkit directory for **all** the target CDB RAC instances.
2. The Golden VDB Pre-Snapshot hook, as provided above, will not work in a clustered (RAC) environment with more than one active instance because it only shuts down the local instance. `dbms_pdb.describe` will not execute while an instance is open read-write. The workarounds are:
   a. Provision the Golden VDB as single-instance, either by provisioning to a non-RAC target or by provisioning to a RAC target with only one active instance. The sample hook will work in this case.
   b. Write a customized pre-snapshot hook that shuts down all instances, restarts only one instance in read-only mode, and runs `dbms_pdb.describe`.
   c. Manually perform the actions of the hook: shutdown the Golden VDB, restart one of the instances in read-only mode, and then run `dbms_pdb.describe`.