Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
**Table of Contents**

**Disclaimer**  

**Oracle Database 12.2 Multitenant Online Provisioning, Refresh and Relocation**  

**Online (Hot) Oracle Database 12.2 Cloning**  

- Oracle Multitenant 12.1 Cloning Recap  
- Oracle Multitenant 12.2 Online (Hot) Cloning  
- Required Configuration  
  - Local UNDO Configured and Enabled  
  - ARCHIVELOG MODE Enabled  
  - Common User and Privileges  
  - Public Database Link *(Optional)*  
- Execution Work Flow  
- Customer Use Case – Continuous Integration and PDB Provisioning  

**Online Oracle Database 12.2 Fast Refresh Synchronization**  

- Required Configuration  
- Refresh Modes  
  - Refresh Mode Manual  
  - Refresh Mode Automatic  
- Execution Workflow  
  - Refresh Mode Manual  
  - Refresh Mode Automatic  
- Customer Use Case – Hybrid Cloud Gold Image Refreshable Clone Operations
Online Oracle Database 12.2 Relocation

Required Configuration 15

TNS Listener Configuration 16

Single TNS LISTENER, Shared Listener Networks or SCAN 16

Independent TNS Listeners 19

Execution Workflow 20

Shared TNS LISTENER Network 20

Isolated TNS LISTENER Networks 21

The Tombstone 23

Customer Use Case – Hybrid Cloud Bi-directional Oracle Database Relocation 23

Conclusion 23
An increasing number of dbPaaS implementations will be done in a hybrid cloud context, which has implications on vendor selection for compatibility, application architecture design and data integration, each of which will need to accommodate both cloud and on-premises environments.”

MARKET GUIDE FOR DATABASE PLATFORM AS A SERVICE
GARTNER ANALYST REPORT: G00281345
15 SEPTEMBER 2016

Oracle Database 12.2 Multitenant Online Provisioning, Refresh and Relocation

Oracle Database 12c Release 2 (12.2), the latest generation of the world’s most popular database, is now available on the Oracle Cloud and for download. As cloud computing continues to gain traction where scalability, elasticity, and cost effective pricing are increasingly important to customers, optimal solutions in database provisioning and database workload management are critical. Core functionality introduced with 12.2 includes enhancements to Oracle Multitenant, specifically in the areas of online database cloning, database clone refresh, and online database relocation. This functionality is available in the Oracle Cloud as well as on-premises environments. This key advantage, offering the exact functionality in the Oracle Cloud and on-premises, allows for consistent execution in both environments and eases the transition to cloud by offering online, hybrid bi-directional provisioning and relocation services between on-premises and the Oracle Cloud.

In Oracle Database 12c Release 1 (12.1), Oracle Multitenant provides a fully integrated, feature rich database provisioning solution using an easily automated, simple command line statement which reduces the time to provision a fully functional Oracle database to minutes. This provisioning mechanism has helped customers build automated private cloud Database-as-a-Service (DBaaS) solutions by cloning from static, gold image databases with ‘baked-in’ customizations relevant to the business. When cloning from active databases for test, development or analytic purposes, however, the 12.1 provisioning required the source database to be read-only during the clone operation. This limitation is removed with Oracle Multitenant in version12.2, where all cloning operations can execute online against ‘hot’ source databases. This expands the DBaaS and time-to-market use cases to include agile, integrated database provisioning to complete a more agile software development lifecycle, micro-services development and continuous application integration and delivery.
Provisioning databases from an active source database introduces a synchronization requirement where the cloned image data deviates from the active source database over time, gradually becoming out of sync. This is problematic when test, development or business analytics require the most recent data. Building on this ‘hot’ cloning capability from an active source database, 12.2 also includes the ability to refresh, either manually or automatically, a gold image database cloned from an active source. The manual refresh option is a simple DDL statement and the frequency of automated refresh is configurable. The advantages of refreshable clones are obvious - development, test and analytics operate on a fresh version of the data. This flexibility, when extended to cloud operations, allows you to keep a cloud-provisioned database in sync with an active, on-premises source database. This in-sync cloud-provisioned database now acts as the clone gold image master database for self-service integrated application development, test or analytic cloud services.

Based on initial customer response, one of the most compelling features in the Multitenant 12.2 release is online database relocation, the ability to move an active database from one hosting environment to another with minimal, or no impact to connected clients. This feature does not require external operating system virtualization, but depends solely on the in-database virtualization inherent to Oracle Multitenant. This database relocation moves only the data relevant to the database itself and eliminates the excess of the operating system image copy. Some relevant use cases of such a feature might be the online relocation of databases for planned maintenance, agile workload balancing of active databases, and bi-directional migrations to and from the Oracle Cloud. In each case, minimizing downtime preserves business continuity. Database relocation can be executed on-premises, or between database services in the Oracle Cloud or as hybrid operations between on-premises databases and database services in the Oracle Cloud.

Oracle Multitenant 12.2 is a highly scalable, in-database virtualization solution where a container database, CDB$ROOT, provides database infrastructure such as CPU utilization, process scheduling, memory utilization, I/O and database resource management to hosted tenants known as pluggable databases (PDBs). These hosted PDBs have their own SYSTEM, SYSAUX, USER and UNDO tablespaces with their associated data files. This level of segregation permits the agility in provisioning and portability described. When coupled with other features such as database lockdown and performance profiles, Oracle Multitenant Application Containers and the ease of hybrid cloud operations, the 12.2 provides a cost-effective, solid foundation for secure DBaaS and SaaS, either on-premises, in the Oracle Cloud or through hybrid operations in both. What follows is a technical description of the provisioning and relocation services with relevant use cases possible with Oracle Multitenant in 12.2.
Online (Hot) Cloning of PDBs

Multitenant 12.1 Cloning Recap

Database provisioning through cloning was available in the first release of Oracle Multitenant 12.1. This version introduced the PDB$SEED, which is default template PDB created solely for fast provisioning of subsequent databases. This simple cloning procedure has reduced what was once a very complicated database provisioning task to a simple DDL statement: `create pluggable database mydb admin user oracle identified by oracle`. A quick recap of 12.1 cloning features is a relevant reminder of the PDB cloning features which continue to be available in 12.2:

- **Full Clone.** This is the creation of a new database as a copy of the source database with complete duplicate of data files. This cloning functionality is available for local or remote. Local full clones are clones executed in the same hosting container database (CDB). Remote full clones are clones executed across hosting container databases (CDBs) either on the same OS/server or different OS/servers respecting endian compatibility.

- **Snapshot Clone.** This is the creation of a new database as a copy of the source database with a snapshot copy of the data files. A snapshot copy leverages a storage vendor’s copy-on-write or similar technology which permits a consistent, thin provisioned copy of the source database while ‘branching’ any changes to blocks as modified on the source or cloned database. Oracle Multitenant snapshot clones can be local or remote. Remote snapshot clones require shared storage. For example, 2 CDBs in a RAC clustered environment sharing common ASM storage.

- **Subset Clone.** True for both full or snapshot clones, database subset clones allow you to create a clone database using only a subset of the source database data set.

- **Metadata Clone.** True for both full or snapshot clones, similar to a database subset clone, a database metadata clone only creates a database clone of the source database metadata excluding the data.

- **On-clone Trigger.** True for all clone types where a database trigger will be defined in the source database and fired in the clone database before the clone database is first opened. This is often used to implement data masking, data redaction or other implemented security features that protect sensitive data in the cloned image where all traces of the security policy are removed from the clone.

In Multitenant 12.1 each of these cloning features required the source database to be transactionally quiesced or cloned from an archived PDB, which has been unplugged. In Oracle Multitenant 12.2 these provisioning tasks are now online, ‘hot’, operations on an active source database.

Online (Hot) Cloning

In 12.1 all hosted pluggable databases in a common container share the same UNDO tablespace. This fact inhibited independent, online database provisioning from an active database. In 12.2, shared UNDO is still available, however, local UNDO, configured for the pluggable database (PDB) is also available and is required for online (‘hot’) cloning. Local UNDO is a property of the CDB$ROOT and when enabled, all hosted tenants for the CDB will have their own local UNDO tablespace. The UNDO tablespace is automatically created when cloning from the PDB$SEED only if the CDB is in local UNDO mode. You cannot mix shared UNDO and local UNDO in the same CDB.

---

1 Generic snapshot copy functionality is available on any file system supporting sparse files and where the oracle init.ora parameter clonedb=true. See My Oracle Support (MOS) Note 1210656.1 for configuration.
Required Configuration

The following configuration is required to enable online database provisioning from an active database:

Local UNDO Configured and Enabled

As mentioned above, when cloning from the PDB$SEED or creating a pluggable database using DBCA an undo tablespace is created by default when local UNDO is enabled for the CDB. Confirm this by executing from CDB$ROOT as sysdba:

```sql
SQL> select p.con_id, p.name, c.tablespace_name
from v$pdb$ p, cdb_tablespaces c
where p.con_id=c.con_id
and tablespace_name like upper('%undo%');
```

```
CON_ID NAME    TABLESPACE_NAME
---------- ------- ------------------
 5 APP1 UNDOTBS1
 4 APP3 UNDOTBS1
 3 APP2 UNDOTBS1
```

**Note:** CDB and PDB local UNDO management is AUTO by default and the default UNDO RETENTION remains 900 seconds. The management and behavior of UNDO tablespaces in the context of a PDB should be identical to the CDB or non-CDB. The PDB$SEED includes an UNDOTBS1 tablespace for default provisioning purposes.

To confirm that local undo is enabled for the CDB, execute as sysdba in CDB$ROOT:

```sql
SQL> select property_name from database_properties where property_name like upper('%local_undo%');
```

```
PROPERTY_NAME
-------------
LOCAL_UNDO_ENABLED
```

In cases where local undo is not enabled for the CDB, execute the following as sysdba in CDB$ROOT:

```sql
SQL> startup upgrade;
SQL> alter database local undo on;
SQL> shutdown immediate;
SQL> startup;
```

**Note:** Again, Local UNDO is enabled in the context of the CDB$ROOT and enabled for all tenants hosted in that container. You cannot mix local and shared UNDO. A PDB with a local UNDO tablespace but without local UNDO enabled will continue to use the shared UNDO tablespace in the CDB$ROOT.
ARCHIVELOG MODE Enabled
In order to satisfy a hot clone request archived logs may be required. To enable archivelog mode in the CDB$ROOT execute the following as sysdba:

```
SQL> startup mount
SQL> alter database archivelog;
SQL> alter database open;
```

**NOTE:** The Flash Recovery Area (FRA) should be adequately sized following 80/20 or 40/60 (DATA/FRA) rules base on storage redundancy requirements.

Common User and Privileges
Database provisioning is executed in the context of the CDB$ROOT as a common user who must have the following minimum privileges:

» Create session, resource. These can be granted to a common role.

» Create pluggable database privilege in both the source and destination CDB$ROOT. The common user in the source database should have either create pluggable database or the SYSOPER admin privilege. These can be granted to a common role.

» Grant SYSOPER container=all to the common user on the destination CDB so that the common user can open the PDB after creation.

An example of the common user privilege definition executed as sysdba in source and destination databases would be:

```
SQL> grant create session, resource to c##clone_admin container=all;
SQL> grant create pluggable database to c##clone_admin container=all;
SQL> grant sysoper to c##clone_admin container=all;
SQL> alter user c##clone_admin set container_data=ALL container=current;
SQL> grant select on cdb_pdbs to c##clone_admin;
```

Public Database Link (Optional)
If you expect to hot clone a database between distinct CDBs, a public database link should be created from the destination CDB to the source CDB connecting as a privileged common user that exists in both environments. For example, as sysdba in the destination CDB$ROOT:

```
SQL> create public database link POD1_link
    connect to c##clone_admin identified by password
    using 'POD1';
```

where c##clone_admin has the privileges defined above and 'POD1' represents a TNS connection alias defined in the tnsnames.ora file.²

---
² If you use such an alias for the database link in a RAC environment, then we require the same tnsnames.ora entry on all RAC instances, because Parallel File Copy will create slave processes on all RAC instances, and they will all attempt to read the dblink description to establish a remote connection to copy the source PDB files.
Moving From Shared UNDO to Local UNDO

When moving from a PDB which was configured for shared undo in the CDB$ROOT or adopting a non-CDB as a PDB into a CDB where local undo is configured and enabled, an UNDO tablespace(s) will be created automatically for the PDB when it is plugged and the UNDO tablespace will be used immediately for that PDB.

Moving from Local UNDO to Shared UNDO

There are situations where you might want to move from the local UNDO configuration to use shared UNDO, for example when the database may be transitioned to read only for archival purposes. In this scenario, you will relocate the PDB to a CDB with shared undo configured using an PDB unplug / plug operation. Once opened, the PDB will use the shared UNDO tablespace in the CDB$ROOT and the local UNDO tablespace can be dropped. An unplugged PDB manifest indicates whether local UNDO is configured for that PDB - `<localundo>1</localundo>`.

Execution Work Flow

Multitenant 12.2 offers online 'hot' clone provisioning which can be executed locally in the context of a single CDB or remotely across CDBs. In either scenario, when storage is shared and the storage supports sparse files and is correctly configured, a hot clone may be thin provisioned using the snapshot copy syntax resulting in faster execution time and less storage consumed. In the following example, Figure 1 below, we assume no shared storage and we are executing a hot clone between 2 CDBs. These CDBs may be in the same data center or different data centers such as an on-premise to Oracle Cloud hybrid operation.

The example in Figure 1 begins with a production CDB with 3 application PDBs represented. The production system is moving forward in time from $T_0 \rightarrow T_{20}$. At $T_{20}$ a hot clone operation is initiated (1) from the development environment. In this release, the hot clone operation is a ‘pull’ executed across the public database link from the destination CDB. A `BEGIN CLONE SCN` marker is set for the hot clone at the beginning of a bulk block copy of the source DATA FILES, REDO LOGs and UNDO from the production CDB for the APP3 PDB to the development CDB. When the file copy is complete, an `END CLONE SCN` is set, at $T_{30}$ in the example below. An ‘alter pluggable database devapp3 open’ is executed (2) for the hot cloned DEVAPP3 PDB. The open statement applies recovery rolling forward changes and undo uncommitted transactions, rendering the hot clone image, DEVAPP3, transactionally consistent as of $T_{30}$, the `END CLONE SCN`.

3 In Real Application Clusters deployments a UNDO tablespace per PDB on each instance where the PDB is deployed is automatically created.

4 Oracle Multitenant 12.2.0.1 supports snapshot copy syntax on ACFS, ZFS, NETAPP, EMC File systems and environments where dNFS/CLONEDB has been configured.
The alert.log for the development CDB will list the media recovery window to satisfy the hot clone as of `BEGIN CLONE SCN` to `END CLONE SCN`:

Applying media recovery for pdb-4099 from SCN `1512265` to SCN `1512315`
...
Successful media recovery should also post to the alter.log:
...
DEVAPP3(4):Incomplete Recovery applied until change `1512315` time 12/19/2016 15:13:49

The `END CLONE SCN`, the point at which the source and destination PDBs begin to deviate, is also exposed in the `CDB_PDBS` view in the context of `CDB$ROOT` or, as in the SQL below, `DBA_PDBS` in the context of the PDB:

```
SQL> select pdb_name, creation_scn from dba_pdb;
```

```
PDB_NAME CREATION_SCN
-------- ---------------
DEVAPP3 1512315
```

As described in the above example, using two simple DDL statements:

```
SQL> create pluggable database devapp3 from app3@dblink parallel 16;
SQL> alter pluggable database devapp3 open;
```

creates a cloned database from an active database source. The data files are physical block copies from the source executed in parallel threads when including the parallel clause in the create statement as indicated above. When
snapshot copy compliant shared storage is available, the hot clone operation is even more efficient in execution time and storage consumed. This is a cost-effective fully integrated Oracle Multitenant database solution that does not require external OS virtualization or extra licensing.

Customer Use Case – Continuous Integration and PDB Provisioning

Swiss Mobiliar presented the following use case at Oracle OpenWorld 2016\(^5\) where they describe their migration to agile, micro service application development integrating Oracle Database 12c Multitenant PDBs as the backend data store. Swiss Mobiliar explored integrating database images such as PostgreSQL and non-CDBs in docker containers for stack provisioning but decided on bare metal Multitenant PDB provisioning as the integrated solution due to performance, efficiency, ease of integration and manageability of the environment. The time to provision the entire stack is less than 10 minutes with the application running in a docker container connecting to the PDB through Oracle Internet Directory (OID) via an EZ connect string. In this provisioning workflow, the Continuous Integration Server orders a new PDB for the application. The PDB is created and the PDB state information along with an OID connect string entry is sent back to the Continuous Integration Server, which then orders the application docker container with the OID entry embedded in the request (see the associated footnote for link details to the presentation).

![Diagram](image)

**Figure 2.** Fully automated application deployment including Oracle database in less than 10 minutes

A detail of the automated PDB provisioning workflow described above is further described in Figure 3. A REST Call initiated from Continuous Integration executes a function over SQLNET to schedule a job that creates the PDB as a clone from a template gold image PDB. A PDB orders table in a PDB provisioning repository is updated once the PDB is provisioned and sends success state back to Continuous Integration that subsequently creates the associated application docker\(^6\).

---

\(^5\) ‘Agile Development, Cost Saving, and More with Oracle’s Multitenancy Solutions’ [CON1668], Alain Fuhrer, Oracle Database Administrator, Mobiliar Versicherungen AG

\(^6\) Ibid.
This use case illustrates the simplicity in provisioning a feature rich Oracle Multitenant 12.2 database and the ease of integration in automated, fully managed, agile development environments.

Online Fast Refresh Synchronization

Oracle Multitenant 12.2 online database cloning creates a cloned image as of the END CLONE SCN established when the data file copy is complete and the PDB is opened. At this point the cloned image begins to deviate from the source and they are out of sync. Without refresh capabilities for the cloned PDB, you would have to drop the first cloned image PDB and recreate it. This would require another copy of the data files, REDO and UNDO to instantiate a new cloned image PDB as of a new END CLONE SCN. This synchronization effort is inefficient for refresh purposes and, once the hot clone operation is complete, we are again out of sync with the source. Oracle Multitenant 12.2 extends the Multitenant online database cloning capabilities by allowing you to define a hot cloned gold image PDB as ‘refreshable’ clone master. The initial hot clone operation is as described in the Online Database Cloning section earlier but includes a refresh mode declarative in the create statement which keeps the hot cloned gold image PDB moving forward in time, in sync with the source PDB.

Required Configuration

The required configuration for Oracle Multitenant 12.2 Database Fast Refresh Synchronization is identical to the online (hot) database cloning requirements defined earlier. An online (hot) cloned PDB is defined as ‘refreshable’ by appending a refresh mode clause to the create pluggable database statement.

Refresh Modes

If you have a requirement to keep the source and cloned image in sync you would provision the cloned image as a refreshable copy using the refresh mode declarative in the create statement. There are two refresh modes available.

Refresh Mode Manual

- refresh mode manual requires the user to initiate the refresh:

  SQL> create pluggable database oedev from oe@dblink refresh mode manual;
Executed as a common user in the CDB$ROOT with the minimum privileges as described in the Common User and Privilege section above, this statement would create a cloned image PDB OEDEV from an active (hot) source PDB OE, in this case in a remote CDB via a database link, using the refresh mode manual declarative which defines this cloned PDB as a refreshable copy where the end user is responsible for the manual execution of the cloned PDB refresh.

Refresh Mode Automatic

- refresh mode automatic implements the refresh using Oracle Scheduler based on a frequency declared in the create pluggable database statement:

```
SQL> create pluggable database oedev from oe@dblink refresh mode every 360 minutes;
```

Again, executed as a common user from CDB$ROOT on the destination CDB assuming the correct privileges defined for that user, this statement would create a cloned image PDB OEDEV in a remote CDB, using the refresh mode declarative refresh mode every 360 minutes.

Execution Workflow

The refreshable clone is intended to keep periodically in sync with a source PDB and act as a gold image clone master from which other PDBs are created as full or snapshot copies. This removes the overhead of multiple, successive cloning operations on the source PDB. The gold image refreshable clone master should not deviate from the source apart from the refresh operation. For this reason, the gold image refreshable clone is mounted in read only mode. As stated, the refresh operation can be one of two types: manual or automatic. In both cases, the gold image refreshable clone master must be closed during the refresh operation. The refresh operation is a physical block apply of REDO and UNDO to render the gold image refreshable clone master transaction state consistent as of an END CLONE SCN, c.f. Online (hot) database cloning. The following two sections describe both refresh modes.

Manual Refresh Mode

Figure 4 below steps through the initial creation of a refreshable clone and the manual refresh operations following an SCN timeline on the source CDB. Step 1 executes the create pluggable database statement on the destination CDB over a database link, from destination to source, at T20 on the SCN timeline. As we have seen with the online database provisioning described earlier, the create statement defines a BEGIN CLONE SCN and copies the source PDB’s REDO, UNDO and DATA FILES to the destination CDB as of the BEGIN CLONE SCN. When the copy completes, the END CLONE SCN is marked as of T30. When the PDB open statement executes, media recovery is applied rendering the PDB DEVAPP3 transaction state consistent as of SCN T30 and in step 2 the refreshable clone is opened in read-only mode consistent as of SCN T30.

Step 3 below initiates the refresh operation. The gold image refreshable clone DEVAPP3 must be closed when the refresh command is executed in the CDB$ROOT where the DEVAPP3 clone is hosted. Step 4 executes the refresh where only the relevant REDO and UNDO is copied based on a new BEGIN CLONE SCN T70. A new END CLONE SCN is established, in this case at T90. Media recovery is applied to a bring DEVAPP3 to a transaction consistent state as of a new END CLONE SCN T90. Step 5 opens the gold image refreshable clone DEVAPP3 is opened read only as of SCN T90.
NOTE: Just as with the online (hot) database cloning, the media recovery window for the `BEGIN CLONE SCN` and the `END CLONE SCN` is posted to the hosting CDB alert.log:

```
DEVAPP1_REFRESH_AUTO(3):alter pluggable database refresh
2017-01-04T13:49:33.282989-05:00
Applying media recovery for pdb-4099 from SCN 1503528 to SCN 1504696
```

Automatic Refresh Mode

The steps to create and refresh a gold image refreshable clone automatically are similar to the manual steps, however the initial create statement defines the refresh mode with an automated refresh frequency rather than manual mode. The automated refresh operation creates a repeating database Scheduler job with the frequency interval defined in the create statement. **Figure 5** below steps through the automated refresh where **step 1** indicates a refresh every 6 hours, defined in minutes, in the create pluggable database statement.
Figure 5. Automatic Gold Image PDB Clone Create and Refresh Operations

The workflow is identical to the manual refresh but is executed as a database scheduled job. The gold image refreshable clone DEVAPP3 must be closed for the scheduled job to complete. The scheduled job can be monitored using the dba_scheduler_jobs view from CDB$ROOT on the CDB hosting the gold image refreshable clone:

```sql
SQL> SELECT job_name, repeat_interval, last_start_date, next_run_date FROM dba_scheduler_jobs WHERE job_name like '%DEVAPP3%';

<table>
<thead>
<tr>
<th>JOB_NAME</th>
<th>REPEAT_INTERVAL</th>
<th>LAST_START_DATE</th>
<th>NEXT_RUN_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEVAPP3_REFRESH_AUTO_3835478423_REFRESH</td>
<td>FREQ = MINUTELY; INTERVAL = 360</td>
<td>04-JAN-17 01.18.03.4</td>
<td>04-JAN-17 07.18.03.3</td>
</tr>
</tbody>
</table>
```

**NOTE:** scheduled jobs can be executed immediately using dbms_scheduler.run_job(), for example:

```sql
exec dbms_scheduler.run_job('DEVAPP3_REFRESH_AUTO_3835478423_REFRESH')
```

**Customer Use Case – Hybrid Cloud Gold Image Refreshable Clone Operations**

Oracle Database 12.2 is available in the Oracle Public Cloud and for download. In an effort to reduce operational and capital costs associated with development environments, customers are transitioning these environments from on-premises LOB deployments or private cloud deployments to hybrid operations provisioning these databases in Oracle Database Cloud Services from on-premises production environments. **Figure 6** below illustrates an easy transition for these environments from on-premises to the Oracle Database Cloud Services leveraging a hybrid refreshable clone-provisioning model. Using the gold image refreshable clone hybrid capabilities coupled with the inherent security in the Oracle Database Cloud Services using baked-in PDB lockdown profiles, TDE, Database Vault and Data Masking features, customers are able to minimize the cost for these environments through metered, fully managed services.
The online (hot) clone operation is a background task that can execute the data file copy in parallel. The initial gold image clone is created as an automatic refreshable clone where the refresh occurs at scheduled intervals and the refresh itself is a REDO apply without the need for a full copy clone. The DEVAPP1 and BI APP1 PDBs are either full clones, snapshot clones or subset clones taken from the DEV GOLD IMAGE refreshable clone residing in the Oracle Public Cloud database service and refreshed from the on-premises source database.

Online Oracle Database 12.2 Relocation

One of the most compelling features in the Oracle Database 12.2 Multitenant release is the ability to relocate an active database from one location to another with minimal or no impact to connected clients. Whether the objective is to achieve levels of consolidation density, workload balancing across servers, bursting on-premises workloads to the cloud or simply satisfying the need to relocate the database for planned maintenance, the agility and ease of database relocation using the Oracle Multitenant 12.2 relocation services is significant. This service builds on the database online provisioning and refresh services described earlier and introduces specific network connection handling for certain TNS network topologies to minimize, or remove entirely, the disruption to the application service.

Required Configuration

The configuration for Oracle 12.2 Multitenant relocation requires the minimum configuration for online database provisioning – local undo configured and enabled, archive log mode configured and enabled, common user with required privileges defined on both source and destination CDBs and a functional public database link configured from destination to source based on either the CDB default service or a managed service for the CDB. The relocate can be executed across different operating systems as long as the endian format is the same. To facilitate the relocation without user intervention, the relocating database must have the same or subset of options available in the destination CDB and be at an identical patch version and the source and destination names in the create statement must be the same. To initiate the database relocation, the command is syntactically similar to any create pluggable database statement but introduces a relocate clause with two modes, based on the SQLNET
listener configuration – *high availability*, the default mode, which presumes shared SQLNET network between source and destination CDBs or *maximum availability*, which requires a declarative that indicates the SQLNET network is not shared and that the client connection forwarding should be handled explicitly.

**SQLNET Listener Configuration**

The following describes TNS LISTENER network topologies that influence the choice of database relocation execution.

**Single TNS LISTENER, Shared Listener Networks or SCAN**

If both the source and destination CDBs participating in the relocation operation share the same TNS LISTENER, for example, a single common listener to which both CDBs are registered, or multiple TNS LISTENERs cross registered to form a LISTENER network, or, a shared SCAN network in an Oracle Grid Infrastructure deployment, related database services may be known to all TNS LISTENERs such that any database relocation and the relocation of associated database services will automatically re-register with the hosting TNS LISTENER and all client connections will be re-directed accordingly with no required change to the client connect string. In this configuration, the correct relocation syntax as executed from the destination CDB$ROOT as the common user would be:

```sql
SQL> create pluggable database APP1 from APP1@dblink relocate;
```

This syntax assumes a shared TNS network where database service registration will occur automatically as part of the database relocation. The following example of a SCAN shared network for 2 CDBs in a clustered environment illustrates the configuration:

```
CDB$ROOT$CDB1-SQL> show parameters listener
NAME TYPE VALUE
------------- --------------
listener_networks string
local_listener string
remote_listener string SCAN-1:1521

CDB$ROOT$CDB1-SQL> show pdbs
CON_ID CON_NAME OPEN MODE RESTRICTED
--------- ----------- -------------
 2 PDB$SEED READ ONLY NO
 3 APP2 READ WRITE NO
 4 APP3 READ WRITE NO
 5 APP1 READ WRITE NO

$oracle@host1-cdb1: lsnrctl services
Service "app1" has 1 instance(s).
  Instance "CDB1", status READY, has 1 handler(s) for this service...
  Handler(s):
    "DEDICATED" established:0 refused:0 state:ready LOCAL SERVER

CDB$ROOT$CDB2-SQL> show parameters listener
NAME TYPE VALUE
------------- --------------
```

---

8 SCAN is an acronym for Single Client Access Name in the Oracle Grid Infrastructure.
In this configuration we see 2 CDBs – CDB1 and CDB2 each sharing a SCAN listener network – SCAN-1:1521. APP1 PDB is hosted on CDB1 and registered with the listener listening on port 1521 on that server. After the relocation of APP1 from CDB1 to CDB2 the APP2 service registration is automatically registered to the new CDB2 instance.

Relocate the PDB APP1 from CDB1 to CDB2:

```sql
CDB$ROOT@CDB2> create pluggable database app1 from app1@cdb1_link relocate;
```

Confirm state of PDB APP1 on CDB2:

```sql
CDB$ROOT@CDB2> show pdbs
```

```
CON_ID CON_NAME OPEN_MODE  RESTRICTED
---------- --------- ------------- ------------
2  PDB$SEED  READ ONLY  NO
3  DEVAPP_REFRESH_AUTO READ ONLY  NO
4  APP1    READ WRITE  NO
```

Confirm the service registration with the listener on host2 for CDB2:

```bash
$ oracle@host2-cdb2: lsnrctl services
Service "app1" has 1 instance(s).
  Instance "CDB2", status READY, has 1 handler(s) for this service...
  Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
      LOCAL SERVER
```

In the above configuration, a shared SCAN network for CDB1 and CDB2, the EZCONNECT string resolves to the SCAN address, SCAN-1:1521/app1 and works implicitly after relocation. Similarly, if there were no SCAN network configured but TNS_LISTENERS were cross-registered to form a common listener network, the client connection would also resolve after relocation.
The following is such an example:

```
CDB$ROOT@CDB1-$SQL> show parameters listener
NAME       TYPE      VALUE
---------- ---------- ---------------------
listener_networks string ((name=network1)(local_listener=LISTENER_CDB1)(remote_listener=LISTENER_CDB2))
local_listener  string  LISTENER_CDB1
remote_listener  string  LISTENER_CDB2

CDB$ROOT@CDB2-$SQL> show parameters listener
NAME       TYPE      VALUE
---------- ---------- ---------------------
listener_networks string ((name=network1)(local_listener=LISTENER_CDB2)(remote_listener=LISTENER_CDB1))
local_listener  string  LISTENER_CDB2
remote_listener  string  LISTENER_CDB1

CDB$ROOT@CDB1-$SQL> show pdbs
CON_ID CON_NAME       OPEN MODE RESTRICTED
--------- ----------- ---------------
2    PDB$SEED         READ ONLY  NO
3    APP2            READ WRITE  NO
4    APP3            READ WRITE  NO
5    APP1            READ WRITE  NO
```

Both CDB1 and CDB2 have local and remote listeners configured and defined in the listener_networks parameter. CDB1 also hosts the PDB APP1. Below we see the service registration for the PDB APP1 with LISTENER_CDB1. Note that the service has multiple service handlers registered across two hosts, in this case on the same subnet, listening on distinct ports – LISTENER_CDB1 host1:1523 and LISTENER_CDB2 host2:1524. An EZCONNECT string of type @host1:1523/app1 or @host2:1524/app1 will resolve the connection due to cross-listener registration.

```
$oracle@host1-cdb1: lsnrctl services
Service "app1" has 1 instance(s).
  Instance "CDB1", status READY, has 6 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:7 refused:0 state:ready
        LOCAL SERVER
      "DEDICATED" established:0 refused:0 state:ready
        REMOTE SERVER
          (ADDRESS=(PROTOCOL=TCP)(HOST=host1.us.oracle.com)(PORT=1523))
      "DEDICATED" established:0 refused:0 state:ready
        REMOTE SERVER
          (ADDRESS=(PROTOCOL=TCP)(HOST=host2.us.oracle.com)(PORT=1524))
```
After the PDB APP1 relocates to HOST2 CDB2 we query the TNS LISTENER service on HOST1 CDB1 and we see it is now hosted in CDB2:

```
$oracle@host1-cdb1: lsnrctl services
Service "app1" has 1 instance(s).
  Instance "CDB2", status READY, has 6 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:7 refused:0 state:ready
        LOCAL SERVER
      "DEDICATED" established:0 refused:0 state:ready
        REMOTE SERVER
          (ADDRESS=(PROTOCOL=TCP)(HOST=host2.us.oracle.com)(PORT=1524))
      "DEDICATED" established:0 refused:0 state:ready
        REMOTE SERVER
          (ADDRESS=(PROTOCOL=TCP)(HOST=host1.us.oracle.com)(PORT=1523))
```

```
CDB$ROOT@CDB2-SQL> show pdbs
 CON_ID CON_NAME OPEN MODE RESTRICTED
    ----- --------------- ---------------
         2 PDB$SEED READ ONLY NO
         3 APP1 READ WRITE NO
```

**NOTE:** A single listener to which both CDB1 and CDB2 have registered is a simpler subset of the shared cross-listener registration use case. Also, using Oracle Connection Manager (CMAN) will also allow TNS LISTENER cross-registration with the back-end TNS LISTENERS.

**Independent TNS Listeners**

When the source and destination CDBs do not share a common TNS LISTENER network and they are completely independent, the TNS network tier must be notified to handle client connection forwarding. Oracle Database 12.2 introduces SQL*Net functionality which embeds a forwarding connection address in the registered service address entry. To effect this change to handle client connection forwarding in this TNS topology the correct relocation syntax as executed from the destination CDB$ROOT as the common user must be:

```
create pluggable database APP1 from APP1@dblink relocate availability max;
```

This syntax, `relocate availability max;` is a declarative indicating that the SQL*NET layer must handle the client connection forwarding when the TNS LISTENER networks are completely isolated. An example of such a topology follows. The database listener configuration identifies only a single local listener in both CDBs:

```
CDB$ROOT@CDB1-SQL> show parameters listener
 NAME TYPE VALUE
------- -------- ---------------
listener_networks string
local_listener string LISTENER_CDB1
remote_listener string
```

```
CDB$ROOT@CDB2-SQL> show parameters listener
```
and where the PDB APP1 is hosted on HOST1 CDB1 as reflected in the service registration with the LISTENER_CDB1:

```bash
$oracle@host1-cdb1: lsnrctl services
Service "app1" has 1 instance(s).
  Instance "CDB1", status READY, has 1 handler(s) for this service...
  Handler(s):
    "DEDICATED" established:0 refused:0 state:ready
LOCAL SERVER
```

The relocate command when issued with the relocate availability max clause will relocate the PDB APP1 to the destination HOST2 CDB2 and will update the listener service definition on the original host CDB1 TNS LISTENER service entry with the following:

```bash
$oracle@host1-cdb1: lsnrctl services
Service "app1" has 1 instance(s).
  Instance "cdb1", status READY, has 1 handler(s) for this service...
  Handler(s):
    "COMMON" established:0 refused:0 state:ready
FORWARD SERVER
  (ADDRESS=(PROTOCOL=TCP)(HOST=host2.us.oracle.com
   (PORT=1524))
```

Execution Workflow

**Shared TNS LISTENER Network**

When there is a shared TNS LISTENER network between the source and destination CDBs participating in the relocation operation, the SQL*Net layer implicitly handles the client connection redirection inherent to cross-registered TNS LISTENERS. Figure 7 below describes this relocation operation.
The APP1 PDB is relocated, online, from CDB1 to CDB2. The CDBs may be collocated on the same server, in different servers in the same cabinet or, in different servers geographically dispersed. The REDO, UNDO and DATA FILE copy is initiated as a background operation while the APP1 PDB is active on CDB1. When the APP1 PDB is opened on CDB2, APP1 PDB on CDB1 is closed; APP1 PDB services are registered with the TNS LISTENER and available on CDB2 when APP1 PDB is opened. The potential client impact is related to the time it takes to execute the ALTER PLUGGABLE DATABASE APP1 OPEN statement. This is when media recovery takes place and the APP1 PDB on CDB2 transaction state is consistent as of the END CLONE SCN described earlier in the online database provisioning section.

NOTE: The FILE COPY progress for all online provisioning operations can be monitored by querying the view v$session_longops:

```
CDB$ROOT@CDB2$SQL> select opname, message from v$session_longops
OPNAME                     MESSAGE
-------------------------------
kpdbfCopyTaskCbk            kpdbfCopyTaskCbk: /u01/app/oracle/oradata/cdb1/CDB : 904448 out of 904448
                               8 Blocks done
```

Isolated TNS LISTENER Networks

In the relocation operation where the TNS LISTENER networks are isolated and the relocate availability max is used, at the time the relocated database is opened on the destination CDB, the TNS LISTENER on the

---

9 There are 2 new status flags in the views cdb_pdb / dba_pdb – RELOCATING and RELOCATED. These status flags may be useful for status diagnostics or to identify whether relocation was executed with the ‘availability max’ clause.
10 The relevant OPNAMES in this view for file copy operations are kpdbfCopyTaskCbk (data file copy) and kcrfremnoc (redo file copy)
source CDB updates the registered service for all services defined for the relocated PDB embedding the forwarding address to the new hosting destination TNS LISTENER, as in the example above, HOST2 listening on port 1524.

As described in Figure 8, in this topology and relocation operation it is important to note the following:

» The relocating PDB remains available on the source hosting CDB until the alter pluggable database APP1 open; is executed on the destination CDB

» The data file copy and iterative redo apply runs in the background and can be executed in advance of the final relocate operation, for example, initiate the relocate on Friday and open the relocated PDB on Sunday before the business week begins.

» This TNS LISTENER topology requires the relocate availability max clause

» When the relocating PDB is opened on the new hosting CDB:
  » listener_networks is updated in the source PDB defining the forwarding address and the listener PDB services on the old hosting CDB TNS LISTENER are updated with the forwarding address
  » internally, the relocated PDB is first opened read only
  » Read-only connections are immediately forwarded to the new hosting TNS LISTENER and new read-write connections are forwarded to the new hosting CDB TNS LISTENER where they spin until the APP PDB is opened in a transaction consistent state.
  » the source PDB executes shutdown immediate and persistent connections are terminated
  » a source PDB artifact referred to internally as a tombstone remains on the original host

Figure 8. Online database relocation with TNS LISTENER connection forwarding

NOTE: When using Oracle Application Continuity11, persistent connections would terminate but their transactions would be replayed, masking the outage and further preserving business continuity.

11 Details regarding Application Continuity with Oracle Database 12c Release 2 can be found at http://www.oracle.com/technetwork/database/database-cloud/private/application-continuity-wp-12c-1966213.pdf
The Tombstone

The preserved tombstone is a PDB artifact unique to the online relocate operation where there is no shared SQLNET LISTENER network and the availability max clause is used. The tombstone preserves the PDB name space in the old hosting CDB so that a PDB of the same name cannot be created in the CDB until the SQLNET LISTENER service connection forwarding is removed. This includes the relocation back to the original hosting CDB while the tombstone is still present. The connection forwarding is a transient state until the client connect strings are updated. As best practice, client connections should be negotiated using Oracle Internet Directory (OID) or some other LDAP functionality to simplify client connection management. Once the connect strings are updated to point to the new hosting CDB SQLNET LISTENER the tombstone artifact can be dropped and any restrictions are removed.12

Customer Use Case – Hybrid Cloud Bi-directional Oracle Database Relocation

There are a number of use cases for hybrid cloud bi-directional database relocation, such as cloud bursting for workload management, patching and upgrade (HW, OS, CDB or PDB), application migration in to the Oracle Cloud, migration of databases between different quality database cloud services located on-premises or in the Oracle Cloud. Figure 8 below illustrates a use case for bi-directional hybrid database relocation operations13 where production, test or development databases ‘burst’ in to the cloud for extra compute resources for immediate or scheduled processing or to temporarily offload workloads to the cloud giving more compute to on-premises workloads.

Mobile application development is another use case where application developers working remotely can easily relocate the latest versions of an application and associated database for testing and further development from the

12 Connection forwarding can be stopped by dropping the Tombstone like a regular PDB

DROP PLUGGABLE DATABASE <pdbname> INCLUDING DATAFILES

13 Please see ‘Hybrid Cloud Agility Begins with a Multitenant Architecture’ presentation describing application durable location transparency and online database relocation services at: https://www.youtube.com/watch?v=uP1aWuzYWmo
cloud and to the cloud. This solution offers a low-cost option facilitating an agile software development life cycle and management.

Conclusion

Oracle Multitenant is a container database solution where the Container Database (CDB) provides database compute infrastructure to hosted tenants called pluggable databases (PDBs) and does not require operating system virtualization. PDBs are self-contained databases for each application with their own SYSTEM, SYSAUX, USER and UNDO tablespaces and associated data files. This autonomy and degree of isolation enables the online portability and provisioning services described in this paper, with little or no impact to the client and no application changes required. The Oracle Multitenant architecture provides for course-grained or fine-grained management strategies where a single privileged user can manage the entire CDB and all hosted tenants performing patching, upgrades and backups as a single entity. Alternatively, administrative tasks can be delegated to privileged users as PDB ADMINs, managing within the context of their respective PDB. The Oracle Multitenant online provisioning and relocation services are fully integrated features and are compatible with Oracle Cloud, customer on-premises, and hybrid-cloud operations.

Oracle Database 12c Release 2 (12.2) is available in the Oracle Cloud (or for download) and offers a simple, low cost entry point to access best of breed, enterprise technology without compromising performance and which provides a quick measure of the return on investment. The management interfaces for the Oracle Cloud are the same as on-premises and do not require new management skill sets. Oracle Multitenant 12.2 is the database architecture behind the Oracle Database Cloud fully managed database services. In addition to the bi-directional database provisioning, refresh and relocation services for intra-cloud, on-premises or hybrid cloud operations discussed in this paper, Oracle Multitenant 12.2 increases PDB high-density consolidation from 256 PDBs per CDB to 4096 hosted PDB tenants in a single CDB, with the continued ability to manage many-as-one for patching, upgrade and high availability. Additional features of the Oracle Database 12c Release 2 relevant to efficient and secure database provisioning include PDB level performance and lockdown profiles which embed default or user configured CPU, I/O and MEMORY database resource management policies and database lockdown security features in the context of individual PDBs. As PDBs are provisioned the performance and security profiles are ‘baked in’ the cloned image and defined and functional when the PDB is opened. Each of these features – hot clone, refreshable clone, database relocate, scalable tenant density with fine-grained CPU, I/O and Memory resource management and in-grained lockdown profiles - are essential to a reliable, secure, highly scalable yet manageable, cloud deployment whether in a public, private or hybrid cloud model.

---

14 Oracle Multitenant is fully compatible with Oracle RAC and Oracle Data Guard and greatly simplifies HA management where all hosted tenants benefit from HA configured at the CDB managing many-as-one.
Integrated Cloud Applications & Platform Services

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. 0116

Oracle Multitenant Database Provisioning – Oracle Database 12c Release 2 (12.2)
January 2017
Author: John P. McHugh, Oracle Database Multitenant Product Management
Contributing Authors: Sanjai Jain, Giridhar Ravipati, Patrick Wheeler, Can Tuzla