ORACLE

Migrating SAP NetWeaver® Based Systems Within the Scope of Oracle Databases

November 2021 (reviewed and verified November 2023) | Version 1.10

Disclaimer

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. 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 in this document remains at the sole discretion of Oracle. Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

Revision History

The following revisions have been made to this document since its initial publication:

DATE	REVISION
November 2023	Reviewed and verified as current
November 2022	Reviewed and verified as current
November 2021	• Expanded the scope of the paper and renamed it from “Migrating SAP NetWeaver Based Systems to Oracle Exadata Cloud Solutions” • Added information about Oracle Multitenant migration
February 2021	Initial version

Table of Contents

Scope and Assumptions

This document outlines the common methods and necessary steps for migrating systems based on SAP NetWeaver within the scope of Oracle Databases. This includes migrations of most SAP systems that currently run on traditional Oracle single-instance non-container databases (non-CDB) to Oracle Multitenant, and systems on customer-specific Oracle Real Application Clusters (RAC) implementations or Oracle engineered systems like Exadata Cloud@Customer or Exadata Cloud Service. This document also complements the following documents:

- SAP NetWeaver Application Server ABAP/Java on Oracle Exadata Cloud@Customer X9M
- SAP NetWeaver Application Server ABAP/Java on Oracle Exadata Cloud@Customer X8M
- SAP NetWeaver Application Server ABAP/Java on Oracle Exadata Cloud Service
- SAP NetWeaver Application Server ABAP/Java on Oracle Cloud Infrastructure

The focus of this document is on Oracle Database–specific questions and tasks in conjunction with SAP NetWeaver. It does not focus on specific aspects of hardware platforms, virtualization solutions, or cloud infrastructure-specific topics (such as setting up a virtual cloud network or security rules).

This document discusses the relevant migration tools and methods, as well as their advantages and disadvantages. Migration methods specific to Oracle Multitenant and Oracle Recovery Manager (RMAN), such as RMAN Back Up, Restore, and Recover Database and RMAN Duplicate Database, are discussed in detail, and examples of most of the required steps are provided. Other migration methods might be described in detail in subsequent versions of this document, depending on and prioritized by customer demands.

Basic Considerations for Database Migrations

When migrating databases, you must consider several things, some more obvious than others. For example:

- Vendor of the source and target database
- Versions and patch levels of the database software
- OS and architecture of the platforms involved
- Encryption
- Different types of storage for the database relevant files
- Oracle RAC–related requirements and configuration changes
- Size of the database

Migrating databases for SAP NetWeaver can be even more complex, and the scope of tasks to perform for the migration can vary widely. For example, migrating an existing traditional SAP NetWeaver system on a traditional single-instance Oracle non-CDB database to Oracle Multitenant without changing the OS platform can be much simpler than migrating the same system to another OS platform or to an Oracle engineered system that runs in Oracle Cloud Infrastructure (OCI).

Following are the most important aspects to consider before choosing your migration methods or services:

- Migration to Oracle Multitenant architecture
- Whether the source environment consists of an Oracle Database on Linux x86_64 and is being migrated from a single instance, from on-premises Exadata, or from a customer-built Oracle RAC database cluster to Exadata Cloud@Customer or Exadata Cloud Service

- Whether the source environment is using an Oracle Database on an OS platform *not* based on Linux x86_64 and requires a heterogenous platform migration
- Whether the source environment must be migrated from a database platform other than Oracle Database
- Downtime considerations and requirements for migration
- The robustness of the migration path
- Data encryption using Transparent Data Encryption (TDE)
- Oracle Database software versions and patch levels
- Modifications to the SAP NetWeaver software, such as RFC connections, scheduled jobs, SAP Solution Manager changes, SAP Gateway changes, SAP Transaction DB13, or new SAP licenses
- Changes in your existing operating concept
- New skills required for your technical staff
- Involvement of external service partners

You must consider what aspects are affected and relevant for *your* migration project.

Database Migration Tools, Methods, and Services

Numerous methods and services exist for migrating a system based on SAP NetWeaver to the different target environments, whether it is a standalone database server, a customer-specific RAC system, or an engineered system like Oracle Exadata Cloud@Customer or Exadata Cloud Service. This section describes those methods and services, and lists pros and cons for each one.

Oracle Multitenant

Starting with Oracle Database 19c, Oracle Multitenant is supported by SAP for single-instance databases, customer-specific RAC databases, and Oracle engineered systems such as on-premises Exadata, Exadata Cloud@Customer, and Exadata Cloud Service.

Not all types of existing SAP configurations qualify for a direct migration from a non-CDB to Oracle Multitenant. One reason is some lack of functionality in tools that support Oracle Multitenant during the initial rollout phase of SAP support for Oracle Multitenant architecture. SAP development is ongoing, and new functionality to benefit more from Oracle Multitenant will be added over time. However, some system combinations, such as SAP Multiple Components on One Database (MCOD), will likely never be supported by SAP in an Oracle Multitenant setup because these combinations might require additional steps when migrating to Oracle Multitenant architecture.

Note: To avoid serious obstacles or an unsupported configuration, follow SAP Note 2336881: Using Oracle Multitenant with SAP NetWeaver based Products. This SAP Note provides a high-level picture on the scope of supported SAP configurations with Oracle Multitenant. Check it for prerequisites and potential limitations regarding the SAP configurations and their qualification to migrate to Oracle Multitenant architecture. The note is updated regularly to reflect any changes in the configuration options, and it provides links to related topics, such as SAP user concepts or mixing SAP Online Analytical Processing (OLAP) and SAP Online Transaction Processing (OLTP) in one CDB.

If you need clarification about other system prerequisites, contact SAP for specific recommendations.

For Oracle Multitenant, SAP Software Provisioning Manager (SWPM) supports the preparation of hosts for CDB-only installations and CDB plus pluggable database (PDB) installations.

Oracle Multitenant offers a smart way to migrate existing SAP databases from a non-CDB to a PDB on the same system or to another—possibly remote—system, if the source and target hosts run on same OS and have the same endianness. A common case is migration from a single-instance non-CDB SAP database to a PDB hosted in a single-instance CDB or to Oracle RAC, for example, on an on-premises Exadata, an Exadata Cloud@Customer, or an Exadata Cloud Service system.

Oracle Multitenant supports multiple ways to create a PDB. For example, you can create a PDB from a remote non-CDB or PDB by using a database link on the target CDB that points to the source non-CDB or PDB. Or you can plug the non-CDB or PDB into an existing CDB by converting it into a PDB by moving or copying the database files to a new directory structure. For more information about both of these methods, see the following sections.

With shared undo tablespaces in the target CDB, the undo tablespaces from the source are automatically replaced in a new PDB, as are the online redo logs. Consider this when deciding about the size of undo tablespaces, undo retention, and the size and number of online redo logs.

Creating a PDB by Using a Database Link to the Source Database

This method creates a PDB in the target CDB by copying the whole source database (non-CDB or PDB) over a network connection. It can transform a single-instance database into an Oracle RAC database without the need to make numerous RAC-specific modifications to the database.

Pros

- This method allows migrations from file system to file system, from file system to Oracle Automatic Storage Management (Oracle ASM), and from Oracle ASM to Oracle ASM.
- Depending on the performance of the network between the source and target, and the storage subsystem and the CPUs, the migration can be highly parallelized.
- TDE-encryption keys are migrated automatically.
- File names and database service names can be mapped to new names.
- SAP SWPM supports this method by providing functionality such as CDB-only installation, CDB plus PDB installation, installation of additional PDBs into an existing CDB, and SAP host preparation.

Cons

- This method does not support heterogenous migrations between different OS platforms.
- Migration happens offline.

Creating a PDB Based on the Manifest of Another PDB or Non-CDB

This method can create a PDB in the target CDB in two ways: by copying the source database (non-CDB or PDB) into a new PDB or by integrating an existing non-CDB or PDB into a running CDB. By copying the whole source database (non-CDB or PDB) on the attached file system or storage, you can keep your original non-CDB or PDB. By integrating the non-CDB into a CDB without copying, you can turn the non-CDB into a PDB. In addition, this method can transform a single-instance database into an Oracle RAC database without the need to make numerous RAC-specific modifications to the database.

Pros

- A PDB can be created on the file system or Oracle ASM as a copy from a non-CDB or PDB. The non-CDB or PDB can also be located on the file system or ASM.

- A PDB can be created by directly integrating and turning the non-CDB into a PDB in the CDB.
- Performance is mostly limited by the performance of the attached file systems or storage subsystems.
- If the PDB is created as a copy, file names and database service names can be mapped to new names.
- If the PDB is created by integrating and turning the non-CDB into a PDB in the CDB, files can be moved into a new directory structure.
- SAP SWPM supports this method by providing functionality such as CDB-only installation, CDB plus PDB installation, installation of additional PDBs into an existing CDB, and SAP host preparation.

Cons

- This method does not support heterogenous migrations between different OS platforms.
- Migration happens offline.

Oracle Recovery Manager

Oracle Recovery Manager (RMAN) supports multiple migration methods, all with different pros and cons.

- Back Up, Restore, and Recover Database
- Duplicate Database
- Transportable Tablespaces
- Full Transportable Export/Import
- Transportable Database

Back Up, Restore, and Recover Database

With the Back Up, Restore, and Recover Database method, the source database is backed up and then restored and recovered on the target system.

Pros

- This method is efficient and fast because it can be highly parallelized.
- This method is easy to set up and run.
- Only a few post-migration tasks are required on the database side.
- With Oracle Database 19c and the appropriate SAP Bundle Patches, the database can be restored and TDE-encrypted in one step.

Cons

- The source and target database software (Oracle Database homes) must be the same version and must have the same SAP Bundle Patches installed.
- The source and target database must be on the same OS and have the same endian type.
- Migration happens offline.
- Shared disk space is required to back up the database on the source system and to restore it on the target system.

Duplicate Database

The Duplicate Database method is similar to the Back Up, Restore, and Recover Database method. However, instead of backing up the database to disk and restoring it from disk, the database is copied directly from source to target through a SQL*Net (network) connection.

Pros

- Because of parallelism, this method can be efficient and fast if network bandwidth is high enough and latency is low.
- It's easy to set up and run.
- Only a few post-migration tasks are required on the database side.
- With Oracle Database 19c and the appropriate SAP Bundle Patches, the database can be restored and TDE-encrypted in one step.
- No shared disk space for backup is required.

Cons

- The source and target database software (Oracle Database homes) must be the same version and must have the same SAP Bundle Patches installed.
- The source and target database must be on the same OS and have the same endian type.
- Migration happens offline. Although the database can be open during duplication, the SAP application should be offline for consistency.
- Speed is limited by the network.

Transportable Tablespaces

The Transportable Tablespaces (TTS) method lets you migrate the data files with application data between different OS platforms and endian types. The data files of the source database must be accessible on the target system (for example, on NFS) and can be converted and copied into Oracle ASM in one step. Other than the methods in which data is exported from the source database and imported into a new target database, TTS is the only supported method for migrating heterogenous platforms without the need to export data or read and write it using the SQL layer.

This method is a combination of converting source data files into target data files and exporting and importing just metadata by using Oracle Data Pump (expdp and impdp).

Typically, only application-specific tablespaces are transported. The target database is created and prepared as a new database with just the SYSTEM and SYSAUX tablespaces by using SAP SWPM. All the tablespaces except SYSTEM and SYSAUX are dropped (with the `including datafiles and contents` clause) after initial creation. Each required tablespace from the source is then migrated into Oracle ASM on the target system, and metadata about those tablespaces is imported using impdp.

Endian conversion can be done by using either RMAN CONVERT or the DBMS_FILE_TRANSFER package. For more information about how to use TTS for migration, see Oracle Database: Migrating Non-CDBs to New Hardware with a Different Endian Operating System and for a New Release.

Pros

- You do not need to export all the data from tables.
- You do not need to re-create indexes or statistics.
- Endian conversion can be parallelized on the data-file level.
- The source and target database software versions can be different as long as the “compatible” initialization parameter is set properly.

Cons

- There is no support for endian conversion and TDE-encryption in one step.
- Tablespaces must be self-contained (locally managed).
- A separate export and import of metadata by using Data Pump is required.
- A new database with just the SYSTEM and SYSAUX tablespaces is required, and the source tablespaces must be imported.
- Migration happens offline.
- The migration is complex and requires many manual steps.
- Migration time is mainly determined by the time it takes to export and import the metadata of the single database user SAPSR3. This operation can be slow because metadata export and import cannot be parallelized, and all SAP data belongs to the single database user SAPSR3 with about 100,000 tables and more than 200,000 indexes. With SAP Business Warehouse, this process is even slower because there are typically several hundreds of thousands or millions of table and index partitions.

Full Transportable Export/Import

Full Transportable Export/Import (FTEX) is based on Transportable Tablespace technology and on Oracle Data Pump, and it provides the functionality of TTS in one command. Data Pump copies the specified data files from the source to the target and then exports and imports the relevant metadata.

The pros and cons of this method are almost the same as with TTS. For more information, see Oracle Database documentation or collateral material, such as the Full Transportable Export and Import technical brief.

Transportable Database

The Transportable Database method supports conversions of whole databases between different OS platforms of the same endian type. Examples are migrations from Microsoft Windows to Linux x86_64 (little endian to little endian) or Solaris SPARC to IBM AIX (big endian to big endian). The conversion of IBM AIX to Linux x86_64 (big endian to little endian) is not supported.

Pros

- You do not need to export all the data from tables.
- You do not need to re-create indexes or statistics.
- Conversion can be parallelized.
- The source and target database software versions can be different as long as the “compatible” initialization parameter is set properly.

- Migration of the whole database (including the SYSTEM and SYSAUX tablespaces) is supported.

Cons

- Endian conversion is not supported.
- TDE-encryption cannot be performed in one step.
- Migration happens offline.
- The migration is complex and requires many manual steps.

Oracle Data Pump

Oracle Data Pump is a toolset that supports multiple migration techniques, all with different pros and cons. The toolset consists of the command line tools impdp and expdp, and the DBMS_DATAPUMP and DBMS_METADATA PL/SQL packages.

Oracle Data Pump can help to move data in the following ways:

- **Data file copying**: This method is basically the Transportable Tablespace method explained in the preceding section. You copy (and convert, if necessary) data files by using Oracle RMAN and export and import tablespace metadata by using expdp and impdp.
- **Direct path**: Usually, direct path unloading and loading of data can greatly improve performance during migrations in which data file copying cannot be used. However, there are numerous situations in which direct path cannot be used. For more information, see Using Direct Path to Move Data.
- **External tables**: Moving data by using external tables does not have the limitations of direct path unless there are conflicting table attributes. For more information, see Using External Tables to Move Data.
- **Conventional path**: The conventional path is based on export and import through the SQL layer, which is also one of the migration methods with the biggest effect on performance.
- **Network link**: When you copy data over a network link (in this context, the same as a database link), the data is copied from the source database to the target database over the network. No export dump files need to be written. This method tries to read and write the data by using direct path; if that is not possible, it uses conventional path. For more information, see Using Network Link Import to Move Data.

Oracle Data Pump requires a pre-existing database on the target system. The data file copying method requires only the SYSTEM and SYSAUX tablespaces to exist in that database. However, the other methods require all the tablespaces to be created as empty tablespaces before data is copied into them.

If you need endian conversion but do not need to export and import data, data file copying (Transportable Tablespace) is the most useful method. Consider the other methods if data needs to be migrated by exporting and importing and you do not want to use SAP R3Load.

Oracle Data Guard Standby Database

Oracle Data Guard provides a set of services that create, maintain, and manage standby databases. Standby databases are full copies of the primary database and can be used to reduce the downtime involved in migrating. Physical standby databases are supported for SAP migration; logical standby databases are not supported.

Following are the pros and cons of using a Data Guard physical standby database for migration.

Pros

- Downtime is minimal.
- The setup is relatively simple.
- You can use this method if the source and target have the same endian type and OS platform.

Cons

- The same database version and patch level are required.
- If the source database is not TDE-enabled, encryption must be enabled as a separate step after migration.
- You cannot use this method for heterogenous migrations between source and target platforms with different endian types.

Oracle Zero Downtime Migration

Oracle Zero Downtime Migration is based on Oracle Data Guard technology and uses a physical standby database to perform online migration. Zero Downtime Migration requires a separate host that orchestrates the whole migration process: building the physical standby database, applying all redo logs from the source, and switching over and swapping roles. Because Zero Downtime Migration is based on physical standby database and supports only Linux x86_64 on the source and target host, only homogeneous migrations are possible. Zero Downtime Migration can transfer initial backup files by using Object Storage, NFS, or Zero Data Loss Recovery Appliance (ZDLRA).

Pros

- We recommend this method for moving databases to Oracle Cloud platforms.
- This method is the same as using Data Guard physical standby. See the previous section for the pros for that method.

Cons

- This method is the same as using Data Guard physical standby. See the previous section for the cons for that method.
- This method has not been successfully tested with SAP.

SAP R3Load

SAP R3Load is one of the most common and well-known methods for SAP database migrations. This method has been optimized over the years and can be used in almost every database migration scenario. With SAP R3load, all SAP-related database tables and views are exported and imported into a prepared and empty target database.

Pros

- SAP R3Load can be used for homogeneous *and* heterogeneous migration scenarios (the OS and database platform can be different).
- SAP R3Load can be used to migrate between different Oracle Database releases.
- SAP R3Load can be highly parallelized on the export and import level.
- Large tables can be split into multiple export files in parallel and imported in parallel.

- Oracle direct path load interfaces are supported for fast data loading.
- Import into TDE-encrypted tablespaces is supported.
- Partitioned tables are supported.
- Reorganization of the entire database is provided, which typically results in space optimization and better performance.
- You can introduce new database features, such as table or index compression or tablespace encryption.

Cons

- Migration happens offline.
- A created target database with appropriately sized encrypted tablespaces is required.
- Only tables defined by SAP can be migrated.
- Triggers cannot be migrated using SAP R3Load.
- Database statistics must be collected after migration.

Migration Services for SAP Offered by Oracle Advanced Customer Services

Oracle offers database migrations as a service and supports customers in all relevant migration scenarios by choosing the approach that works best to meet individual customer requirements. Oracle Advanced Customer Services offers advanced and highly optimized migration methods such as Oracle to Oracle (O2O), Oracle to Oracle Online (Triple O), and Oracle GoldenGate.

Summary of Migration Tools, Methods, and Services

Considering the pros and cons of each method, we recommend the following methods for migrating existing SAP databases:

For homogeneous migrations:

- Oracle Multitenant architecture
- Oracle Data Guard standby database using physical standby
- Oracle RMAN restore or duplicate commands
- Oracle migration services for SAP (GoldenGate, O2O, or Triple O)

For heterogeneous migrations:

- SAP R3Load
- Oracle migration services for SAP (GoldenGate, O2O, or Triple O)

The following sections focus on the Oracle Multitenant and Oracle RMAN migration methods. Other methods might be described in upcoming updates of this document or are already covered in documents and notes available from Oracle or SAP.

Performing Database Migration

This section describes how to use the Oracle Multitenant; Oracle RMAN Backup, Restore, and Recover; and Oracle RMAN Duplicate Database methods to migrate existing SAP databases. Also refer to SAP Note 3018983, which contains complementary information, best practices, and example scripts.

Supported Character Sets

Only the following database character sets and national character sets are supported with SAP:

- For Unicode SAP databases:
  - UTF-8 as the database character set
  - UTF-8 as the national character set
- For non-Unicode SAP databases:
  - WE8DEC as the database character set
  - UTF-8 as the national character set

The migrated target database must use the same database character set and national character set as the source database. Migration using Oracle Multitenant or Oracle RMAN methods do not support changing character sets.

Customers who need to migrate from non-Unicode to Unicode databases must perform an SAP Unicode migration, which is a separate migration project.

Premigration Considerations

Depending on your specific implementation of SAP NetWeaver, you might need to perform some tasks before you can begin the migration. For example:

- Any satellite systems that are connected to the SAP system might require changes.
- Database links from or to the database might also need to be migrated.
- Some SAP jobs might need to be set on hold before migration, so that they do not run into errors when the system is first started up after migration.

Migrating a Database to Oracle Multitenant Architecture

All migration methods based on Oracle Multitenant architecture require a properly set up SAP environment and a CDB in which the new PDB is created. SAP SWPM must be used to prepare the database host or, in the case of Oracle RAC, all the database hosts involved. This creates the SAP-specific OS users and environments.

At the time this document was written, SAP SWPM supported the following Oracle Multitenant–specific installation options:

- Install a new CDB and a new PDB in one step
- Install an additional new PDB into an existing CDB
- Create a CDB without a creating a PDB (CDB-only)

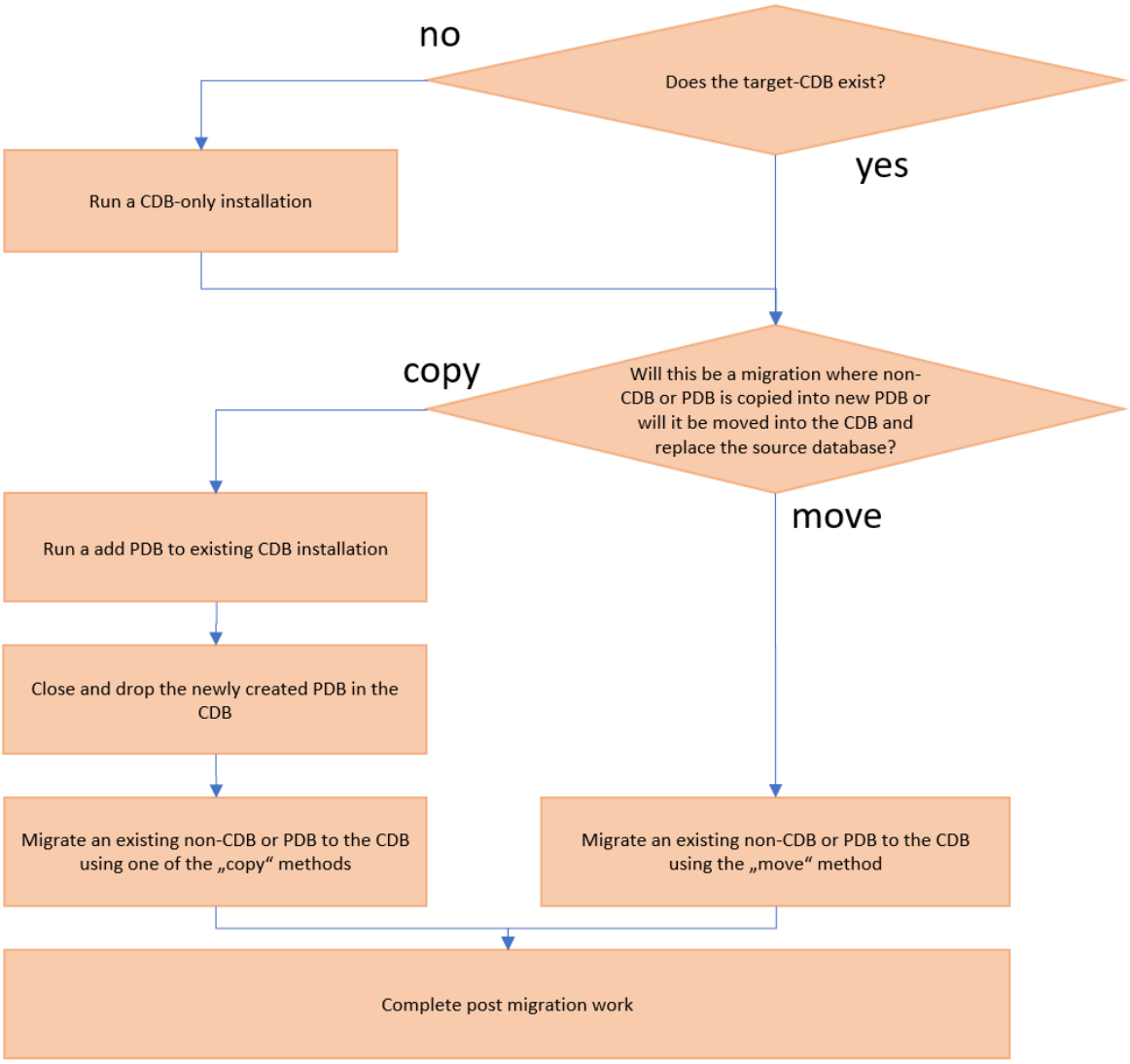
The generic SWPM host-preparation functionality must be used to create SAP-specific OS users and environments on all database hosts, especially for Oracle RAC-based systems where all nodes must be prepared.

New functionality might be added to SWPM by SAP in the future.

SWPM cannot set up a new SAP database instance (profiles, extraction of SAP binaries, SQL*Net configuration, BR*Tools, and so on) without creating the PDB and loading it using R3load. As a result, you must install new SAP database instances, drop the PDB, and replace the dropped PDB with the non-CDB or PDB that has to be migrated.

If you want to stay on the same system with the database, you can use the "move" method by using the `move` clause, which plugs the old non-CDB or PDB into a CDB. If you do not want to touch the old non-CDB or PDB or want to migrate to another system, you can use one of the "copy" methods, either by using the `copy` clause or by copying over a database link. Both methods require additional post-migration steps to adjust the SAP system to the new CDB or PDB environment. Note that the "move" method moves all database-relevant files into the new directory and file structure of the target CDB.

The following illustration gives an overview of the tasks required for the "move" and "copy" methods.



The locations of the required functions in SAP SWPM are as follows:

- SAP SWPM host preparation is located at **Generic Options > Oracle > Preparations > Operating System Users and Groups**.
- SAP SWPM CDB-only installation is located at **Generic Options > Oracle > Database Tools > Create Container DB (CDB) only**.

- To perform a new CDB plus PDB or a new PDB installation into an existing CDB, choose the **Database Instance** option under the **Standard System**, **Distributed System**, or **High Availability System** installation type, depending on your requirements.

General Aspects and Prerequisites

This section provides general information and prerequisites for migration methods based on Oracle Multitenant architecture.

Terminology When Using Oracle Multitenant with SAP

Oracle and SAP have introduced some new terms, such as "single tenant," to distinguish the different ways that an SAP database can be run. These terms are discussed in SAP Note 2336881.

Supported Migration Scenarios

Oracle Multitenant architecture offers several migrations from a source database to a target database:

Non-CDB/single instance	→	Single tenant or multitenant/single instance
Non-CDB/RAC	→	Single tenant or multitenant/RAC
Non-CDB/TDE/single instance	→	Single tenant or multitenant/TDE/single instance
Non-CDB/TDE/RAC	→	Single tenant or multitenant/TDE/RAC
Single tenant or multitenant/single instance	←→	Single tenant or multitenant/RAC
Single tenant or multitenant/TDE/single instance	←→	Single tenant or multitenant/TDE/RAC

Implement the Oracle Standard User Concept

An important and mandatory task when migrating to Oracle Multitenant architecture is converting existing SAP database installations that use the "SAP Classic" user concept to the "Oracle Standard" user concept.

The Oracle Standard user concept adheres more to Oracle recommendations regarding Maximum Availability Architecture (MAA) and Optimal Flexible Architecture (OFA). Moving in this direction, SAP installations are streamlined among different installation types. It is a proven layout, used in all SAP installations with Oracle RAC, Oracle ASM, Oracle Cloud, Oracle engineered systems, and now also for all on-premises installations.

Note: All releases of Oracle Database later than 19c will no longer support non-CDBs. With the transition from traditional non-CDBs to Oracle Multitenant architecture with CDBs hosting multiple PDBs, SAP is switching to the new Oracle Standard user concept for Oracle Databases. The SAP Classic user concept will no longer be supported for Oracle Multitenant architecture.

Although it is not a hard limitation or technical obstacle *not* to convert existing installations of a classic SAP database setup with a single-instance non-CDB, it will significantly reduce the potential risk for problems with the operating environment and the amount of work to do when converting the existing non-CDB into a PDB for use within a CDB.

Following are some of the benefits of converting to the Oracle Standard user concept for Oracle Database with SAP NetWeaver:

- Adjustment in SAP system configuration is uncoupled from conversion to Oracle PDB setup.
- Conversion to the Oracle Standard user concept can be tested long before database conversion is started.
- Smooth transition to use of Oracle Database services with SAP can already be implemented and tested in a non-CDB environment.

Detailed instructions and technical background are provided in the following SAP Notes:

SAP NOTE	DESCRIPTION
SAP Note 1915323 - OS User Concept for SAP NetWeaver for 12c and higher	Explains the difference between the SAP Classic and Oracle Standard user concepts.
SAP Note 1554661 - Configuration of environment for 'oracle' user	Explains how to set up the OS user `oracle` as the owner of all Oracle software used by SAP.
SAP Note 1915317 - Migrating Software Owner to 'oracle'	Provides a script that helps to migrate the ownership of the database data files to the new software owner `oracle`. As a reference, this note also mentions Oracle tools for cloning or migrating Oracle Database home locations to the new software owner.
SAP Note 1983457 - Oracle Home Cloning on Unix and Linux	Explains all potential cloning mechanisms for switching to software owner `oracle`. The note provides Perl scripts for the actual migration of an existing Oracle Database home. It also provides guidance on differences in various Oracle software versions.

Change of ownership of Oracle data files and Oracle Database home must be accomplished in one step.

Note: If you are migrating an existing non-CDB to a CDB that is running on another system (host and user environment) that was already installed with “SAP Classic” as the user concept, you can use the migration approach that copies the source non-CDB to a PDB over a database link, described later. This lets you implement the Oracle Standard user concept during migration and without changes on the source system.

Adopt SAP Naming Conventions of Oracle Database Files on Oracle Multitenant

Usually, SAP database installations with Oracle follow SAP-specific naming rules for all directories and files that belong to an SAP database instance. When you are migrating to Oracle Multitenant, these naming rules change because they need to cover cases with multiple PDBs.

During migration using one of the Oracle Multitenant methods, the traditional non-CDB naming schema must be translated to the naming schema for Oracle Multitenant environments (CDB/PDB). This means that each subdirectory and each data file that corresponds to the non-CDB gets the DBSID of the PDB with an underscore as a prefix, as shown in the following example. If one of the “copy” methods is used, the conversion of the names can be done during copy. If the “move” method is used, the database files must be moved to their new location to match the new structure.

NON-CDB DIRECTORY STRUCTURE AND FILES	CDB/PDB DIRECTORY STRUCTURE AND FILES

[oracle@migtest1 NCD]$ pwd
/oracle/NCD
[oracle@migtest1 NCD]$ tree sapdata*
sapdata1
  cntrl
  cntrlNCD.dbf
  sysaux_1
  sysaux.data1
  system_1
  system.data1
  temp_1
  temp.data1
  undo_1
  undo.data1
sapdata2
  sr3_1
  sr3.data1
sapdata3
  sr3750x.data1
  sr3750x.data1
sapdata4
  sr3usr_1
  sr3usr.data1
	[oracle@migtest1 CDB]$ pwd
	/oracle/CDB
	[oracle@migtest1 CDB]$ tree ncd*
	ncd_sapdata1
	  ncd_sysaux_1
	  ncd_sysaux.data1
	  ncd_system_1
	  ncd_system.data1
	  ncd_temp_1
	  ncd_temp.data1
	  ncd_undo_1
	  ncd_undo.data1
	ncd_sapdata2
	  ncd_sr3_1
	  ncd_sr3.data1
	ncd_sapdata3
	  ncd_sr3750x.data1
	  ncd_sr3750x.data1
	ncd_sapdata4
	  ncd_sr3usr_1
	  ncd_sr3usr.data1

Note: If you are migrating from file system to file system, all the directories in the target must be pre-created as the `oracle:oinstall` user.

If a non-CDB is migrated from a file system to Oracle ASM, you do not need to care about directory structures and file names. Oracle generates the required directory names and files automatically. Each PDB has a unique identifier generated into the file names, as shown in the following example.

NON-CDB DIRECTORY STRUCTURE AND FILES	CDB/PDB DIRECTORY STRUCTURE AND FILES ON ASM

[oracle@migtest1 NCD]$ pwd
/oracle/NCD
[oracle@migtest1 NCD]$ tree sapdata*
sapdata1
  cntrl
  cntrlNCD.dbf
  sysaux_1
  sysaux.data1
  system_1
  system.data1
  temp_1
  temp.data1
  undo_1
  undo.data1
sapdata2
  sr3_1
  sr3.data1
sapdata3
  sr3750x.data1
  sr3750x.data1
sapdata4
  sr3usr_1
  sr3usr.data1
	+DATAC1/CDB/CA75FDF7C56B6C72E053BA01000AF1F4/DATAFILE/system.443.1083166173
	
	+DATAC1/CDB/CA75FDF7C56B6C72E053BA01000AF1F4/DATAFILE/sysaux.447.1083166173
	
	+DATAC1/CDB/CA75FDF7C56B6C72E053BA01000AF1F4/DATAFILE/psapundo.442.1083166173
	
	+DATAC1/CDB/CA75FDF7C56B6C72E053BA01000AF1F4/DATAFILE/psapsr3.447.1083166173
	
	+DATAC1/CDB/CA75FDF7C56B6C72E053BA01000AF1F4/DATAFILE/psapsr3750x.446.1083166173

Note: For all migrations using Oracle Multitenant, use parameter `FILE_NAME_CONVERT` to translate non-CDB-specific file names to CDB/PDB environment-specific file names.

Introduce or Adopt Database Services

With Oracle Multitenant, you must use database services. The tasks required to introduce or adopt database services depend on the type of the source and target systems.

When you are migrating a non-CDB SAP database running as a single-instance database, SAP SWPM has configured Oracle SQL*Net to use an SID to connect SAP application servers to the database instance. This is the default for this type of installation. With Oracle Multitenant, SAP application servers must use database services to connect to the database instead of specifying a SID. For single-instance databases (single tenant or multitenant), the database services must be configured using the `DBMS_SERVICE` PL/SQL package. For Oracle RAC databases, the database services must be configured within Oracle Clusterware using `SRVCTL`.

As a rule, `SERVICE_NAME` and `NET_SERVICE_NAME` are identical, and each SAP application server has its own dedicated database service whose name is a combination of `<SAPSID>` and `<INSTANCE_NAME>`. It might be necessary to deviate from this rule if you are consolidating a larger number of SAP systems into one CDB and some of the SAP systems have the same or overlapping instance names.

For single-instance PDB databases, database services are configured and started as shown in the following example:

```
SQL> exec dbms_service.create_service(´<SERVICE_NAME>´,´<NET_SERVICE_NAME>´);
SQL> exec dbms_service.start_service(´<SERVICE_NAME>´);
```

For Oracle RAC PDB databases, database services are configured and started as shown in following example:

```
svrctl create service -db <CDB> -service <SERVICE_NAME> -pdb <PDB> -preferred <CDB>001 -available
<CDB>002
svrctl start service -db <CDB> -service <SERVICE_NAME>
```

For example, if SAP system `C11` has two SAP application server instances, `C11_DVEBMGS00` and `C11_DV01`, the respective commands for creating and starting the according services in a CDB called `CDB` are as follows:

For a single instance in the PDB `C11`:

```
SQL> exec dbms_service.create_service(´C11_DVEBMGS00´,´C11_DVEBMGS00´);
SQL> exec dbms_service.start_service(´C11_DVEBMGS00´);
SQL> exec dbms_service.create_service(´C11_DV01´,´C11_DV01´);
SQL> exec dbms_service.start_service(´C11_DV01´);
```

For RAC:

```
svrctl create service -db CDB -service C11_DVEBMGS00 -pdb C11 -preferred CDB001 -available CDB002
svrctl start service -db CDB -service C11_DVEBMGS00
svrctl create service -db CDB -service C11_DV01 -pdb C11 -preferred CDB001 -available CDB002
svrctl start service -db CDB -service C11_DV01
```

In addition, the `tnsnames.ora` file must be changed according to the patterns shown in the following table, where an SAP application server uses the SAPSID to connect to the database in traditional non-CDB systems, and a dedicated database service is used for each SAP application server in CDB/PDB environments.

TNSNAMES.ORA IN NON-CDB	TNSNAMES.ORA IN CDB/PDB
<SAPSID>.WORLD = (DESCRIPTION = (ADDRESS LIST = (ADDRESS = (PROTOCOL = TCP) (HOST = <host>) (PORT = <listener port>))) (CONNECT_DATA = (SID = <SAPSID>) (GLOBAL_NAME = <SAPSID>.WORLD)))	# - Entry for SAP instance <INSTANCE_NAME> from SAP system <SAPSID> # - Instance profile has variable # SETENV_xx = dbs_ora_tnsname = <SAPSID>_<INSTANCENAME> set # - Domain is given in file sqlnet.ora # - Used for all types of database the same way: # NON-CDB SI, NON-CDB RAC, CDB, CDB RAC <SAPSID>_<INSTANCE_NAME>.WORLD = (DESCRIPTION = (ADDRESS LIST = (ADDRESS = (PROTOCOL = TCP) (HOST = <host>) (PORT = <listener port>))) (CONNECT_DATA = (SID = <SAPSID>) (GLOBAL_NAME = <SAPSID>.WORLD)))

Finally, the `dbs_ora_tnsname` environment variable must be adjusted in the environments of the standard SAP users `ora<sid>` and `<sid>adm` so that all SAP programs connect to the database using a SQL*Net service name instead of the SID. Although `dbs_ora_tnsname` takes precedence over the same parameter in the SAP instance profiles of each SAP application server, the profiles should be modified accordingly, so that no differences exist between the environment variable and the instance profile.

Choose the next free number for the `SETENV_<##>` statement in the instance profile and set it as follows:

```
SETENV_<##> = dbs_ora_tnsname=<NET_SERVICE_NAME>
dbs/ora/tnsname=<NET_SERVICE_NAME>
```

For example, if the next free number for ## is 06, set `SETENV_06` for SAP instances `C11_DVEBMGS00` and `C11_DV01` and adjust the profiles as follows:

```
SETENV_06 = dbs_ora_tnsname=C11_DVEBMGS00
dbs/ora/tnsname=C11_DVMGS00
SETENV_06 = dbs_ora_tnsname=C11_DV01
dbs/ora/tnsname=C11_DV01
```

Upgrade SAP Bundle Patches and Timezone Files

Although Oracle and SAP are working to support AutoUpgrade, it was not released by SAP when this paper was published. So, you must follow traditional upgrade paths before migration. It is not supported to plug a non-CDB or PDB from an earlier Oracle Database major release into a CDB without first upgrading to the major release version.

SAP Bundle Patch installation already covers Oracle Multitenant configurations. The preferred approach is to install the same SAP Bundle Patch in the source non-CDB or PDB as is installed in the target CDB or to run `catsbp` after creating the PDB. If you want to migrate an existing non-CDB or PDB SAP database to a new PDB, SAP does not support migrating a non-CDB or PDP to a CDB with an earlier version of the SAP Bundle Patch installed.

For more information about restrictions imposed by SAP Bundle Patch levels on the database migration process, see SAP Note 3110260 - Oracle: SAP Bundle Patches and Oracle Database Migration.

Time zone files are another important factor with Oracle Multitenant. Each CDB and PDB has its own time zone file, so you must upgrade each time zone file on each CDB and PDB independently. There is Oracle tool support to help to upgrade the time zone files of the CDB and PDBs as one job. We recommend that you test this upgrade on a nonproduction system first to avoid unexpected obstacles.

See SAP Note 2072655 – FAQ: Oracle Timezone/DST in SAP systems.

Example Migrations Using Oracle Multitenant

In the example workflows shown in the next two sections, it is expected that shell commands and SQL commands are run as the appropriate OS user (for example, `oracle`) and that the Oracle environment variables ORACLE_SID and ORACLE_HOME have been set properly. Always run `. oraenv.sh` and then set both variables to the required values.

For a single-instance non-CDB with the SID `NCD` on the *source*, the typical values are as follows:

- `ORACLE_SID=NCD` (`ORACLE_SID=<SID>`)
- `ORACLE_HOME=/oracle/NCD/19` (`ORACLE_HOME=/oracle/<SID>/19`)

For an Oracle RAC non-CDB with the SID `NCD` on the *source*, the typical values for the first database instance are as follows:

- `ORACLE_SID=NCD001` (`ORACLE_SID=<SID>+<THREAD>`)
- `ORACLE_HOME=/oracle/NCD/19` (`ORACLE_HOME=/oracle/<SID>/19`)

For a single-instance CDB with the SID `CDB` on the *target*, the typical values are as follows:

- `ORACLE_SID=CDB` (`ORACLE_SID=<SID>`)
- `ORACLE_HOME=/oracle/CDB/19` (`ORACLE_HOME=/oracle/<SID>/19`)

For an Oracle RAC CDB with the SID `CDB` on the *target*, the typical values for the first database instance are as follows:

- `ORACLE_SID=CDB001` (`ORACLE_SID=<SID>+<THREAD>`)
- `ORACLE_HOME=/oracle/CDB/19` (`ORACLE_HOME=/oracle/<SID>/19`)

Migrating a Non-CDB to a PDB by Using a Database Link

This section provides the steps for migrating an existing non-CDB to a PDB by copying the database over a database link. The steps, provided in the following table, cover multiple scenarios with one example.

- Sources (in the FROM column) are single-instance (SI) or Oracle RAC non-CDBs.
- Targets (in the TO column) are single-instance (SI) or Oracle RAC PDBs.

You can distinguish between the steps relevant for a specific scenario by paying attention to the FROM and TO columns in the tables.

The example shows how to migrate a non-CDB called `NCD` to a PDB called `NCD`. The CDB on the target hosts, one for single instance and one for Oracle RAC, is called `CDB` in both cases.

Important: The migration assumes that there is a CDB and PDB (which will be dropped before starting migration) configured on the target using SWPM, as described earlier.

FROM	TO	STEP
SI or RAC	SI or RAC	Drop the temporary PDB on the target system. `SQL> drop pluggable database NCD including datafiles;`

SI or RAC	SI or RAC	Create a user on the source database for cloning the non-CDB into a new PDB from remote.
		`SQL> create user mycloneuser identified by "<password>";`
		`SQL> GRANT CREATE SESSION, CREATE PLUGGABLE DATABASE TO mycloneuser;`

SI	SI	Shut down the source database and reopen it in read-only mode.
		`SQL> shutdown immediate;`
		`SQL> startup open read only;`

RAC	RAC	Shut down the source database and reopen it in read-only mode.
		`srvctl stop database -db NCD -stopoption immediate`
		`srvctl start database -db NCD -startoption “READ ONLY”`

SI or RAC on a shared file system	SI or RAC on a shared file system	On the source database, prebuild a `FILE_NAME_CONVERT` clause for later use.
		`SQL> select name from v$datafile;`
		`NAME`
		`----------`
		`/oracle/NCD/sapdata1/system_1/system.data1`
		`/oracle/NCD/sapdata1/sysaux_1/sysaux.data1`
		`/oracle/NCD/sapdata1/undo_1/undo.data1`
		`/oracle/NCD/sapdata2/sr3_1/sr3.data1`
		`/oracle/NCD/sapdata3/sr3750x_1/sr3750x.data1`
		`/oracle/NCD/sapdata4/sr3usr_1/sr3usr.data1`
		
		`SQL> select name from v$tempfile;`
		`NAME`
		`----------`
		`/oracle/NCD/sapdata1/temp_1/temp.data1`
		
		`FILE_NAME_CONVERT=(`
		`'/oracle/NCD/sapdata1/system_1/system.data1','/oracle/CDB/ncd_sapdata1/ncd_system_1/ncd_system.data1',`
		`'/oracle/NCD/sapdata1/sysaux_1/sysaux.data1','/oracle/CDB/ncd_sapdata1/ncd_sysaux_1/ncd_sysaux.data1',`
		`'/oracle/NCD/sapdata2/sr3_1/sr3.data1','/oracle/CDB/ncd_sapdata2/ncd_sr3_1/ncd_sr3.data1',`
		`'/oracle/NCD/sapdata3/sr3750x_1/sr3750x.data1','/oracle/CDB/ncd_sapdata3/ncd_sr3750x_1/ncd_sr3750x.data1',`
		`'/oracle/NCD/sapdata4/sr3usr_1/sr3usr.data1','/oracle/CDB/ncd_sapdata4/ncd_sr3usr_1/ncd_sr3usr.data1',`
		`'/oracle/NCD/sapdata1/temp_1/temp.data1','/oracle/CDB/ncd_sapdata1/ncd_temp_1/ncd_temp.data1',`
		`'/oracle/NCD/sapdata1/undo_1/undo.data1','/oracle/CDB/ncd_sapdata1/ncd_undo_1/ncd_undo.data1'`
		`)`

FROM	TO	STEP

SI or RAC on a file system	RAC on ASM	On the source database, prebuild a `FILE_NAME_CONVERT` clause for later use.
		`SQL> select name from v$datafile;`
		`NAME`
		`----------`
		`/oracle/NCD/sapdata1/system_1/system.data1`
		`/oracle/NCD/sapdata1/sysaux_1/sysaux.data1`
		`/oracle/NCD/sapdata1/undo_1/undo.data1`
		`/oracle/NCD/sapdata2/sr3_1/sr3.data1`
		`/oracle/NCD/sapdata3/sr3750x_1/sr3750x.data1`
		`/oracle/NCD/sapdata4/sr3usr_1/sr3usr.data1`
		
		`SQL> select name from v$tempfile;`
		
		`NAME`
		`----------`
		`/oracle/NCD/sapdata1/temp_1/temp.data1`
		
		`FILE_NAME_CONVERT=(`
		`'/oracle/NCD/sapdata1/system_1/system.data1','+DATAC1',`
		`'/oracle/NCD/sapdata1/sysaux_1/sysaux.data1','+DATAC1',`
		`'/oracle/NCD/sapdata2/sr3_1/sr3.data1','+DATAC1',`
		`'/oracle/NCD/sapdata3/sr3750x_1/sr3750x.data1','+DATAC1',`
		`'/oracle/NCD/sapdata4/sr3usr_1/sr3usr.data1','+DATAC1',`
		`'/oracle/NCD/sapdata1/temp_1/temp.data1','+DATAC1',`
		`'/oracle/NCD/sapdata1/undo_1/undo.data1','+DATAC1')`

SI or RAC on a shared file system	SI or RAC on a shared file system	On the source database, use the names of the data files to prepare the commands to create the required directory structure on the target system.
		`mkdir -p /oracle/CDB/ncd_sapdata1/ncd_system_1`
		`mkdir -p /oracle/CDB/ncd_sapdata1/ncd_sysaux_1`
		`mkdir -p /oracle/CDB/ncd_sapdata2/ncd_sr3_1`
		`mkdir -p /oracle/CDB/ncd_sapdata3/ncd_sr3750x_1`
		`mkdir -p /oracle/CDB/ncd_sapdata4/ncd_sr3usr_1`
		`mkdir -p /oracle/CDB/ncd_sapdata1/ncd_temp_1`
		`mkdir -p /oracle/CDB/ncd_sapdata1/ncd_undo_1`

SI or RAC on a shared file system	SI or RAC on a shared file system	As the `oracle:oinstall` user, create those directories on the target system.
		`CMD> mkdir -p /oracle/CDB/ncd_sapdata1/ncd_system_1`
		`CMD> mkdir -p /oracle/CDB/ncd_sapdata1/ncd_sysaux_1`
		`CMD> mkdir -p /oracle/CDB/ncd_sapdata2/ncd_sr3_1`
		`CMD> mkdir -p /oracle/CDB/ncd_sapdata3/ncd_sr3750x_1`
		`CMD> mkdir -p /oracle/CDB/ncd_sapdata4/ncd_sr3usr_1`
		`CMD> mkdir -p /oracle/CDB/ncd_sapdata1/ncd_temp_1`
		`CMD> mkdir -p /oracle/CDB/ncd_sapdata1/ncd_undo_1`

SI or RAC	SI or RAC	On the target system, add the following lines to the `tnsnames.ora` file and verify that you can connect to the source database.
		`Modify $ORACLE_HOME/network/admin/tnsnames.ora and add:`
		`NCD.WORLD =`
		`(DESCRIPTION =`
		`(ADDRESS = (PROTOCOL = TCP)(HOST = <sourcehost>)(PORT = <sourceport>))`

FROM	TO	STEP
		(CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = ncd))) CMD> sqlplus mycloneuser/"<password>"@NCD
SI or RAC	SI or RAC	On the target CDB, create a database link for the PDB creation from the remote non-CDB. SQL> CREATE DATABASE LINK new_pdb_from_noncdb CONNECT TO mycloneuser IDENTIFIED BY "<password>" USING 'ncd';
SI	SI or RAC on a shared file system	On the target CDB, create the PDB from the remote non-CDB. In the example, the source non-CDB and target CDB have TDE wallets configured. If you are not using TDE, omit the keystore clause. SQL> create pluggable database NCD from NCD@new_pdb_from_noncdb file_name_convert=('/oracle/NCD/sapdata1/system_1/system.data1','/oracle/CDB/ncd_sapdata1/ncd_system_1/ncd_system.data1', '/oracle/NCD/sapdata1/sysaux_1/sysaux.data1','/oracle/CDB/ncd_sapdata1/ncd_sysaux_1/ncd_sysaux.data1', '/oracle/NCD/sapdata2/sr3_1/sr3.data1','/oracle/CDB/ncd_sapdata2/ncd_sr3_1/ncd_sr3.data1', '/oracle/NCD/sapdata3/sr3750x_1/sr3750x.data1','/oracle/CDB/ncd_sapdata3/ncd_sr3750x_1/ncd_sr3750x.data1', '/oracle/NCD/sapdata4/sr3usr_1/sr3usr.data1','/oracle/CDB/ncd_sapdata4/ncd_sr3usr_1/ncd_sr3usr.data1', '/oracle/NCD/sapdata1/temp_1/temp.data1','/oracle/CDB/ncd_sapdata1/ncd_temp_1/ncd_temp.data1', '/oracle/NCD/sapdata1/undo_1/undo.data1','/oracle/CDB/ncd_sapdata1/ncd_undo_1/ncd_undo.data1') KEYSTORE IDENTIFIED BY "<keystorepassword>" PARALLEL 32;
SI	RAC on ASM	On the target CDB, create the PDB from the remote non-CDB. In the example, the source non-CDB and target CDB have TDE wallets configured. If you are not using TDE, omit the keystore clause. SQL> create pluggable database NCD from NCD@new_pdb_from_noncdb file_name_convert=('/oracle/NCD/sapdata1/system_1/system.data1','+DATAC1', '/oracle/NCD/sapdata1/sysaux_1/sysaux.data1','+DATAC1', '/oracle/NCD/sapdata2/sr3_1/sr3.data1','+DATAC1', '/oracle/NCD/sapdata3/sr3750x_1/sr3750x.data1','+DATAC1', '/oracle/NCD/sapdata4/sr3usr_1/sr3usr.data1','+DATAC1', '/oracle/NCD/sapdata1/temp_1/temp.data1','+DATAC1', '/oracle/NCD/sapdata1/undo_1/undo.data1','+DATAC1') KEYSTORE IDENTIFIED BY "keystorepassword" PARALLEL 32;
RAC	RAC	If the source non-CDB and the target CDB are both RAC and on the same cluster, stop and remove the database, instance, and service resources. CMD> srvctl stop database -db NCD -stopoption=immediate CMD> srvctl remove database -db NCD -force

FROM	TO	STEP

SI	SI or RAC	On the target CDB, finalize the new PDB.
		`SQL> alter pluggable database ncd open;`
		`Warning: PDB altered with errors.`
		
		`cd $ORACLE_HOME/sapbundle`
		`CMD> ./catsbp`
		
		`SQL> alter session set container = NCD;`
		`SQL> @?/rdbms/admin/noncdb_to_pdb.sql`
		Verify that all errors and warning have been resolved. If any unresolved errors or warnings exist, resolve them before starting SAP application servers.
		`SQL> set linesize 300`
		`SQL> set pagesize 2000`
		`SQL> col name format a20`
		`SQL> col message format a80`
		`SQL> col cause format a40`
		`SQL> select name,cause,message,status from pdb_plug_in_violations where status <> 'RESOLVED';`
		`no rows selected`
		
		`SQL> alter pluggable database NCD close immediate;`
		`Pluggable database altered.`
		Now the PDB should open without errors or warnings.
		`SQL> alter pluggable database ncd open;`
		`Pluggable database altered.`

SI	SI	On the target system, create the necessary database services so that SAP work processes can connect to the PDB.
		`SQL> alter session set container=NCD;`
		`SQL> exec dbms_service.create_service('NCD_DVEBMGS00','NCD_DVEBMGS00');`
		`SQL> exec dbms_service.create_service('NCD_DV01','NCD_DV01');`
		`SQL> exec dbms_service.start_service('NCD_DVEBMGS00');`
		`SQL> exec dbms_service.start_service('NCD_DV01');`
		`SQL> alter pluggable database save state;`
		`SQL> alter system register;`
		Verify that all the services are registered at the listener. Service "ncd" is registered automatically.
		`CMD> lsnrctl status`
		`---cut---`
		`Service "NCD_DV01" has 1 instance(s).`
		`  Instance "CDB", status READY, has 1 handler(s) for this service...`
		`Service "NCD_DVEBMGS00" has 1 instance(s).`
		`  Instance "CDB", status READY, has 1 handler(s) for this service...`
		`Service "ncd" has 1 instance(s).`
		`  Instance "CDB", status READY, has 1 handler(s) for this service...`
		`---cut---`

FROM	TO	STEP

SI or RAC	RAC	On the target system, use `srvctl` to create the necessary database services in Oracle Clusterware so that SAP work processes can connect to the PDB through the SCAN listener.
		`CMD> srvctl add service -db CDB -service NCD_DVEBMGS00 -pdb NCD -preferred "CDB001" -available "CDB002"`
		`CMD> srvctl add service -db CDB -service NCD_DV01 -pdb NCD -preferred "CDB002" -available "CDB001"`
		`CMD> srvctl start service -db CDB -service NCD_DVEBMGS00`
		`CMD> srvctl start service -db CDB -service NCD_DV01`

SI	SI	On the target system and all your SAP application servers, change the `tnsnames.ora` file so that `NCD.WORLD` resolves to the new PDB instead of the old non-CDB, and add entries for all relevant database services. The typical locations are `<SAPPROF>/oracle/tnsnames.ora`, `<TNS_ADMIN>/tnsnames.ora`, and `$ORACLE_HOME/network/admin`.
		`NCD.WORLD =`
		`  (DESCRIPTION =`
		`  (ADDRESS = (PROTOCOL = TCP)(HOST = <newdbhost>)(PORT = <sourceport>))`
		`  (CONNECT_DATA =`
		`  (SERVER = DEDICATED)`
		`  (SERVICE_NAME = ncd)`
		`  )`
		`  )`
		
		`NCD_DVEBMGS00.WORLD =`
		`  (DESCRIPTION =`
		`  (ADDRESS = (PROTOCOL = TCP)(HOST = <newdbhost>)(PORT = <sourceport>))`
		`  (CONNECT_DATA =`
		`  (SERVER = DEDICATED)`
		`  (SERVICE_NAME = NCD_DVEBMGS00)`
		`  )`
		`  )`
		
		`NCD_DV01.WORLD =`
		`  (DESCRIPTION =`
		`  (ADDRESS = (PROTOCOL = TCP)(HOST = <newdbhost>)(PORT = <sourceport>))`
		`  (CONNECT_DATA =`
		`  (SERVER = DEDICATED)`
		`  (SERVICE_NAME = NCD_DV01)`
		`  )`
		`  )`

FROM	TO	STEP

SI	RAC	On the target system and all your SAP application servers, change the `tnsnames.ora` file so that `NCD.WORLD` resolves to the new PDB instead of the old non-CDB, and add entries for all relevant database services. The typical locations are `<SAPPROF>/oracle/tnsnames.ora`, `<TNS_ADMIN>/tnsnames.ora`, and `$ORACLE_HOME/network/admin`.
		`NCD.WORLD =`
		`  (DESCRIPTION =`
		`  (ADDRESS = (PROTOCOL = TCP)(HOST = <scanlistener>)(PORT = <1521>))`
		`  (CONNECT_DATA =`
		`  (SERVER = DEDICATED)`
		`  (SERVICE_NAME = ncd)`
		`  )`
		`  )`
		
		`NCD_DVEBMGS00.WORLD =`
		`  (DESCRIPTION =`
		`  (ADDRESS = (PROTOCOL = TCP)(HOST = <scanlistener>)(PORT = <1521>))`
		`  (CONNECT_DATA =`
		`  (SERVER = DEDICATED)`
		`  (SERVICE_NAME = NCD_DVEBMGS00)`
		`  )`
		`  )`
		
		`NCD_DV01.WORLD =`
		`  (DESCRIPTION =`
		`  (ADDRESS = (PROTOCOL = TCP)(HOST = <scanlistener>)(PORT = <1521>))`
		`  (CONNECT_DATA =`
		`  (SERVER = DEDICATED)`
		`  (SERVICE_NAME = NCD_DV01)`
		`  )`
		`  )`

SI or RAC	SI or RAC	Modify the environment of all relevant `<sid>adm` users on all SAP application servers, and adjust the SAP instance profiles (`SETENV_##`) of all SAP application server instances as explained earlier, to point to the database services recently created.
		`ncdadm 2> env`
		`SAPSYSTEMNAME=NCD`
		`DIR_LIBRARY=/usr/sap/NCD/SYS/exe/run`
		`RSEC_SSFS_DATAPATH=/usr/sap/NCD/SYS/global/security/rsecssfs/data`
		`RSEC_SSFS_KEYPATH=/usr/sap/NCD/SYS/global/security/rsecssfs/key`
		`rsdb_ssfs_connect=1`
		`LD_LIBRARY_PATH=/usr/sap/NCD/SYS/exe/run:/usr/sap/NCD/SYS/exe/uc/linuxx86_64:/orac`
		`le/client/19/instantclient`
		`SAPEXE=/usr/sap/NCD/SYS/exe/run`
		`THREAD=NOPS`
		`DB_SID=NCD`
		`dbms_type=ORA`
		`dbs_ora_tnsname=NCD_DVEBMGS00`
		`dbs_ora_schema=SAPSR3`
		`ORACLE_SID=NCD`
		`TNS_ADMIN=/usr/sap/NCD/SYS/profile/oracle`
		`ORACLE_HOME=/oracle/NCD/19`

FROM	TO	STEP
		NLS_LANG=AMERICAN_AMERICA.UTF8 SAPDATA_HOME=/oracle/NCD ORACLE_BASE=/oracle/NCD
SI or RAC	SI or RAC	Check SQL*Net connectivity to the new PDB for all relevant SAP application servers and database services. • On the database servers: CMD> sqlplus sapsr3/'<password>'@NCD_DVEBMGS00 CMD> sqlplus sapsr3/'<password>'@NCD_DV01 • On the application servers as <sid>adm: CMD> R3trans -d
SI or RAC	SI or RAC	Create the SAP-specific users and roles in the PDB by running the following PL/SQL scripts in the CDB. The scripts are shipped with BR*Tools. Always use the latest version of BR*Tools. CMD> sqlplus /nolog @sapdba_role NCD CMD> sqlplus /nolog @sapconn_role NCD CMD> sqlplus /nolog @sapuprof_profile NCD

Migrating a Non-CDB to a PDB by Using a Manifest File

This section provides the steps for migrating an existing non-CDB to a PDB by using a manifest XML file. As mentioned in an earlier section, the `file_name_convert` parameter is used to map file names.

You can create the target PDB in the following ways:

- By using the `copy` clause, which copies the relevant files of the source database into a target directory structure on a file system or ASM and makes the copied files members of the new PDB. The `copy` clause leaves the source database unchanged.
- By using the `move` clause, which moves the relevant files of the source database into a target directory structure on a file system or ASM and makes them members of the new PDB.

The steps, provided in the following table, cover multiple scenarios with one example.

- Sources (in the FROM column) are single-instance (SI) or Oracle RAC non-CDBs.
- Targets (in the TO column) single-instance or Oracle RAC PDBs.

You can distinguish between the steps relevant for a specific scenario by paying attention to the FROM and TO columns in the tables.

The example shows how to migrate a non-CDB called `NCD` to a PDB called `NCD`. The CDB on the target hosts, one for single instance and one for Oracle RAC, is called `CDB` in both cases.

Important: The migration assumes that a CDB and a PDB (which will be dropped before starting migration) are configured on the target using SWPM, as described earlier.

FROM	TO	STEP
SI or RAC	SI or RAC	Drop the temporary PDB on the target system. `SQL> drop pluggable database NCD including datafiles;`

SI	SI	Shut down the source database and reopen it in read-only mode.
		`SQL> shutdown immediate;`
		`SQL> startup open read only;`

RAC	RAC	Shut down the source database and reopen it in read-only mode.
		`srvctl stop database -db NCD -stopoption immediate`
		`srvctl start database -db NCD -startoption "READ ONLY"`

SI or RAC	SI or RAC	On the source database, create an XML manifest file. `SQL> exec dbms_pdb.describe('/oracle/NCD/ncd.xml');`

SI or RAC on a file system	SI or RAC on a file system	If the source database is TDE-encrypted, export the encryption keys and encrypt the export file with a separate password.
		`SQL> administer key management export encryption keys with secret "<exportpassword>" TO '/tmp/ncd_tdekey.exp' identified by "<walletpassword>";`
		If this step fails with an "ORA-28417: password-based keystore is not open" error, follow this procedure:
		To successfully export the encryption keys, the password wallet must be open. To open the password wallet, the autologin wallet must be deleted. After you export the encryption keys, you must re-create the autologin wallet. See MOS note 1944507.1, which describes a similar problem.
		Note: We *strongly recommend* that you back up your password wallet and autologin wallet before you delete the autologin wallet.
		1. Get the location of the autologin wallet, and delete it.
		`SQL> show parameters wallet_root`
		`SQL> !rm /oracle/NCD/orawallet/tde/cwallet.sso`
		2. Close the autologin wallet, and open password wallet.
		`SQL> administer key management set keystore close;`
		`SQL> administer key management set keystore open identified by "<walletpassword>";`
		3. Export the encryption keys, and encrypt the export file with a separate password.
		`SQL> administer key management export encryption keys with secret "<exportpassword>" TO '/tmp/ncd_tdekey.exp' identified by "<walletpassword>";`
		4. Re-create the autologin wallet.
		`SQL> administer key management create AUTO_LOGIN keystore from keystore identified by "<walletpassword>";`

FROM	TO	STEP

RAC on ASM	RAC on ASM	If the source database is TDE-encrypted, export the encryption keys and encrypt the export file with a separate password.
		`SQL> administer key management export encryption keys with secret "<exportpassword>" TO '/tmp/ncd_tdekey.exp' identified by "<walletpassword>";`
		If this step fails with an "ORA-28417: password-based keystore is not open" error, follow this procedure:
		To successfully export the encryption keys, the password wallet must be open. To open the password wallet, the autologin wallet must be deleted. After you export the encryption keys, you must re-create the autologin wallet. See MOS note 1944507.1, which describes a similar problem.
		Note: We *strongly recommend* that you back up your password wallet and autologin wallet before you delete the autologin wallet.
		1. Get location of the autologin wallet.
		`SQL> show parameters wallet_root`
		2. (For the removal of the autologin wallet only) Switch the user to `grid` or `oracle` and set ORACLE_HOME to the Grid Infrastructure home and ORACLE_SID to the SID of your ASM instance. Then, `asmcmd` to delete the autologin wallet.
		`ASMCMD> rm +DATAC1/NCD/orawallet/tde/cwallet.sso`
		3. Close the autologin wallet, and open the password wallet.
		`SQL> administer key management set keystore close;`
		`SQL> administer key management set keystore open identified by "<walletpassword>";`
		4. Export the encryption keys, and encrypt the export file with a separate password.
		`SQL> administer key management export encryption keys with secret "<exportpassword>" TO '/tmp/ncd_tdekey.exp' identified by "<walletpassword>";`
		5. Re-create the autologin wallet.
		`SQL> administer key management create AUTO_LOGIN keystore from keystore identified by "<walletpassword>";`

SI or RAC on a shared file system	SI or RAC on a shared file system	On the source database, prebuild a `FILE_NAME_CONVERT` clause for later use.
		`SQL> select name from v$datafile;`
		`NAME`
		`/oracle/NCD/sapdata1/system_1/system.data1`
		`/oracle/NCD/sapdata1/sysaux_1/sysaux.data1`
		`/oracle/NCD/sapdata1/undo_1/undo.data1`
		`/oracle/NCD/sapdata2/sr3_1/sr3.data1`
		`/oracle/NCD/sapdata3/sr3750x_1/sr3750x.data1`
		`/oracle/NCD/sapdata4/sr3usr_1/sr3usr.data1`
		`SQL> select name from v$tempfile;`
		`NAME`
		`/oracle/NCD/sapdata1/temp_1/temp.data1`
		`FILE_NAME_CONVERT=(`
		`'/oracle/NCD/sapdata1/system_1/system.data1','/oracle/CDB/ncd_sapdata1/ncd_system_1/ncd_system.data1',`

FROM	TO	STEP
		'/oracle/NCD/sapdata1/sysaux_1/sysaux.data1','/oracle/CDB/ncd_sapdata1/ncd_sysaux_1/ncd_sysaux.data1', '/oracle/NCD/sapdata2/sr3_1/sr3.data1','/oracle/CDB/ncd_sapdata2/ncd_sr3_1/ncd_sr3.data1', '/oracle/NCD/sapdata3/sr3750x_1/sr3750x.data1','/oracle/CDB/ncd_sapdata3/ncd_sr3750x_1/ncd_sr3750x.data1', '/oracle/NCD/sapdata4/sr3usr_1/sr3usr.data1','/oracle/CDB/ncd_sapdata4/ncd_sr3usr_1/ncd_sr3usr.data1', '/oracle/NCD/sapdata1/temp_1/temp.data1','/oracle/CDB/ncd_sapdata1/ncd_temp_1/ncd_temp.data1', '/oracle/NCD/sapdata1/undo_1/undo.data1','/oracle/CDB/ncd_sapdata1/ncd_undo_1/ncd_undo.data1')
SI or RAC on a file system	RAC on ASM	On the source database, prebuild a FILE_NAME_CONVERT clause for later use. SQL> select name from v$datafile; NAME ---------- /oracle/NCD/sapdata1/system_1/system.data1 /oracle/NCD/sapdata1/sysaux_1/sysaux.data1 /oracle/NCD/sapdata1/undo_1/undo.data1 /oracle/NCD/sapdata2/sr3_1/sr3.data1 /oracle/NCD/sapdata3/sr3750x_1/sr3750x.data1 /oracle/NCD/sapdata4/sr3usr_1/sr3usr.data1 SQL> select name from v$tempfile; NAME ---------- /oracle/NCD/sapdata1/temp_1/temp.data1 FILE_NAME_CONVERT=('/oracle/NCD/sapdata1/system_1/system.data1','+DATAC1', '/oracle/NCD/sapdata1/sysaux_1/sysaux.data1','+DATAC1', '/oracle/NCD/sapdata2/sr3_1/sr3.data1','+DATAC1', '/oracle/NCD/sapdata3/sr3750x_1/sr3750x.data1','+DATAC1', '/oracle/NCD/sapdata4/sr3usr_1/sr3usr.data1','+DATAC1', '/oracle/NCD/sapdata1/temp_1/temp.data1','+DATAC1', '/oracle/NCD/sapdata1/undo_1/undo.data1','+DATAC1')
SI or RAC on a shared file system	SI or RAC on a shared file system	On the source database, use the names of the data files to *prepare* the commands to create the required directory structure on the target system. mkdir -p /oracle/CDB/ncd_sapdata1/ncd_system_1 mkdir -p /oracle/CDB/ncd_sapdata1/ncd_sysaux_1 mkdir -p /oracle/CDB/ncd_sapdata2/ncd_sr3_1 mkdir -p /oracle/CDB/ncd_sapdata3/ncd_sr3750x_1 mkdir -p /oracle/CDB/ncd_sapdata4/ncd_sr3usr_1 mkdir -p /oracle/CDB/ncd_sapdata1/ncd_temp_1 mkdir -p /oracle/CDB/ncd_sapdata1/ncd_undo_1

FROM	TO	STEP

SI or RAC on a shared file system	SI or RAC on a shared file system	As the `oracle:oinstall` user, *create* those directories on the target system and ensure that the source database is mounted under the original mount point (for example, `/oracle/NCD/...`). The PDB is created by copying the original database from those directories.
		`CMD> mkdir -p /oracle/CDB/ncd_sapdata1/ncd_system_1`
		`CMD> mkdir -p /oracle/CDB/ncd_sapdata1/ncd_sysaux_1`
		`CMD> mkdir -p /oracle/CDB/ncd_sapdata2/ncd_sr3_1`
		`CMD> mkdir -p /oracle/CDB/ncd_sapdata3/ncd_sr3750x_1`
		`CMD> mkdir -p /oracle/CDB/ncd_sapdata4/ncd_sr3usr_1`
		`CMD> mkdir -p /oracle/CDB/ncd_sapdata1/ncd_temp_1`
		`CMD> mkdir -p /oracle/CDB/ncd_sapdata1/ncd_undo_1`
		Ensure that the manifest XML file is available at `/oracle/NCD/ncd.xml` and the file with the exported keys is available at `/tmp/ncd_tdekey.exp`.

SI or RAC	SI or RAC	On the target CDB, close and drop the PDB if it exists from earlier tests.
		`SQL> alter pluggable database ncd close instances=all;`
		`SQL> drop pluggable database ncd including datafiles;`

SI or RAC	SI or RAC	On the target CDB, verify whether the non-CDB or PDB can be plugged in.
		`SQL> set serveroutput on`
		`SQL> BEGIN`
		`  IF DBMS_PDB.CHECK_PLUG_COMPATIBILITY( pdb_descr_file => '/oracle/NCD/ncd.xml', pdb_name => 'NCD')=True THEN`
		`    DBMS_OUTPUT.PUT_LINE('DB is compatible');`
		`  ELSE`
		`    DBMS_OUTPUT.PUT_LINE('DB is not compatible');`
		`  END IF;`
		`END;`
		`/`
		
		`DB is compatible`
		Check for warnings or errors that would occur during plug-in operation. (Most of the warnings can be ignored and are resolved when SQL scripts `catsbp` and `noncdb_to_pdb` are run.)
		`set linesize 200`
		`column message format a50`
		`column status format a10`
		`column type format a10`
		`column name format a10`
		`select name, type, message, status from PDB_PLUG_IN_VIOLATIONS where status<>'RESOLVED' order by name,time;`

SI	SI or RAC on a shared file system	On the target CDB, create the PDB by using the manifest XML file. Create the PDB from the non-CDB or PDB by using either the `copy` option or the `move` option.
		`SQL> create pluggable database NCD using '/oracle/NCD/ncd.xml' copy | move file_name_convert=(`
		`'/oracle/NCD/sapdata1/system_1/system.data1','/oracle/CDB/ncd_sapdata1/ncd_system_1/system.data1',`
		`'/oracle/NCD/sapdata1/sysaux_1/sysaux.data1','/oracle/CDB/ncd_sapdata1/ncd_sysaux_1/sysaux.data1',`
		`'/oracle/NCD/sapdata2/sr3_1/sr3.data1','/oracle/CDB/ncd_sapdata2/ncd_sr3_1/sr3.data1',`

FROM	TO	STEP
		'/oracle/NCD/sapdata3/sr3750x_1/sr3750x.data1','/oracle/CDB/ncd_sapdata3/ncd_sr3750x_1/sr3750x.data1', '/oracle/NCD/sapdata4/sr3usr_1/sr3usr.data1','/oracle/CDB/ncd_sapdata4/ncd_sr3usr_1/sr3usr.data1', '/oracle/NCD/sapdata1/temp_1/temp.data1','/oracle/CDB/ncd_sapdata1/ncd_temp_1/temp.data1', '/oracle/NCD/sapdata1/undo_1/undo.data1','/oracle/CDB/ncd_sapdata1/ncd_undo_1/undo.data1') KEYSTORE IDENTIFIED BY "<keystorepassword>" PARALLEL 64;
SI	RAC on ASM	On the target CDB, create the PDB by using the manifest XML file. SQL> create pluggable database NCD using '/oracle/NCD/ncd.xml' copy file_name_convert=('/oracle/NCD/sapdata1/system_1/system.data1','+DATAC1', '/oracle/NCD/sapdata1/sysaux_1/sysaux.data1','+DATAC1', '/oracle/NCD/sapdata2/sr3_1/sr3.data1','+DATAC1', '/oracle/NCD/sapdata3/sr3750x_1/sr3750x.data1','+DATAC1', '/oracle/NCD/sapdata4/sr3usr_1/sr3usr.data1','+DATAC1', '/oracle/NCD/sapdata1/temp_1/temp.data1','+DATAC1', '/oracle/NCD/sapdata1/undo_1/undo.data1','+DATAC1') KEYSTORE IDENTIFIED BY "<keystorepassword>" PARALLEL 64;
RAC	RAC	If the source non-CDB and the target CDB are both RAC and on the same cluster, stop and remove the database, instance, and service resources. CMD> srvctl stop database -db NCD -stopoption=immediate CMD> srvctl remove database -db NCD -force
SI	SI or RAC	On the target CDB, finalize the new PDB. SQL> alter pluggable database ncd open; Warning: PDB altered with errors. If the source database is TDE-encrypted and you exported TDE encryption keys earlier, import the encryption keys. SQL> alter session set container = NCD; SQL> ADMINISTER KEY MANAGEMENT IMPORT ENCRYPTION KEYS WITH SECRET "<exportpassword>" FROM '/tmp/ncd_tdekey.exp' FORCE KEYSTORE IDENTIFIED BY "<walletpassword>" with backup; Convert the non-CDB to the PDB. SQL> @?/rdbms/admin/noncdb_to_pdb.sql Update SAP Bundle Patch to the one installed in CDB. CMD> cd $ORACLE_HOME/sapbundle CMD> ./catsbp SQL> alter pluggable database NCD close immediate instances=all; SQL> alter pluggable database NCD open; Verify that all errors and warning have been resolved. If there are any unresolved errors or warnings, resolve them before starting SAP application servers. SQL> set linesize 300 SQL> set pagesize 2000

FROM	TO	STEP
		SQL> col name format a20 SQL> col message format a80 SQL> col cause format a40 SQL> select name,cause,message,status from pdb_plug_in_violations where status <> 'RESOLVED'; no rows selected

SI	SI	On the target PDB, create the necessary database services so SAP that work processes can connect to the PDB.
		SQL> alter session set container=NCD;
		SQL> exec dbms_service.create_service('NCD_DVEBMGS00','NCD_DVEBMGS00');
		SQL> exec dbms_service.create_service('NCD_DV01','NCD_DV01');
		SQL> exec dbms_service.start_service('NCD_DVEBMGS00');
		SQL> exec dbms_service.start_service('NCD_DV01');
		SQL> alter pluggable database save state;
		SQL> alter system register;
		Verify that all the services are registered at the listener. The "ncd" service is registered automatically.
		CMD> lsnrctl status
		---cut---
		Service "NCD_DV01" has 1 instance(s).
		  Instance "CDB", status READY, has 1 handler(s) for this service...
		Service "NCD_DVEBMGS00" has 1 instance(s).
		  Instance "CDB", status READY, has 1 handler(s) for this service...
		Service "ncd" has 1 instance(s).
		  Instance "CDB", status READY, has 1 handler(s) for this service...
		---cut---

SI	RAC	On the target system, use srvctl to create the necessary database services in Oracle Clusterware so that SAP work processes can connect to the PDB through the SCAN listener. CMD> srvctl add service -db CDB -service NCD_DVEBMGS00 -pdb NCD -preferred "CDB001" -available "CDB002" CMD> srvctl add service -db CDB -service NCD_DV01 -pdb NCD -preferred "CDB002" -available "CDB001" CMD> srvctl start service -db CDB -service NCD_DVEBMGS00 CMD> srvctl start service -db CDB -service NCD_DV01

SI	SI	On the target system and all your SAP application servers, change the tnsnames.ora file so that NCD.WORLD resolves to the new PDB instead of the old non-CDB, and add entries for all relevant database services. The typical locations are <SAPPROF>/oracle/tnsnames.ora, <TNS_ADMIN>/tnsnames.ora, and $ORACLE_HOME/network/admin.
		NCD.WORLD =
		  (DESCRIPTION =
		  (ADDRESS = (PROTOCOL = TCP)(HOST = <newdbhost>)(PORT = <sourceport>))
		  (CONNECT_DATA =
		  (SERVER = DEDICATED)
		  (SERVICE_NAME = ncd)
		  )
		  )
		
		NCD_DVEBMGS00.WORLD =

FROM	TO	STEP
		(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = <newdbhost>)(PORT = <sourceport>)) (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = NCD_DVEBMGS00))) NCD_DV01.WORLD = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = <newdbhost>)(PORT = <sourceport>)) (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = NCD_DV01)))
SI	RAC	On the target system and all your SAP application servers, change the `tnsnames.ora` file so that `NCD.WORLD` resolves to the new PDB instead of the old non-CDB, and add entries for all relevant database services. The typical locations are `<SAPPROF>/oracle/tnsnames.ora`, `<TNS_ADMIN>/tnsnames.ora`, and `$ORACLE_HOME/network/admin`. NCD.WORLD = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = <scanlistener>)(PORT = <1521>)) (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = ncd))) NCD_DVEBMGS00.WORLD = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = <scanlistener>)(PORT = <1521>)) (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = NCD_DVEBMGS00))) NCD_DV01.WORLD = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = <scanlistener>)(PORT = <1521>)) (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = NCD_DV01)))
SI/RAC	SI/RAC	Modify the environment of all relevant `<sid>adm` users on all SAP application servers, and adjust the SAP instance profiles (`SETENV_##`) of all SAP application server instances, as explained earlier, to point to the database services recently created. ncdadm 2> env SAPSYSTEMNAME=NCD

FROM	TO	STEP
		DIR_LIBRARY=/usr/sap/NCD/SYS/exe/run RSEC_SSFS_DATAPATH=/usr/sap/NCD/SYS/global/security/rsecssfs/data RSEC_SSFS_KEYPATH=/usr/sap/NCD/SYS/global/security/rsecssfs/key rsdb_ssfs_connect=1 LD_LIBRARY_PATH=/usr/sap/NCD/SYS/exe/run:/usr/sap/NCD/SYS/exe/uc/linuxx86_64:/oracle/client/19/instantclient SAPEXE=/usr/sap/NCD/SYS/exe/run THREAD=NOPS DB_SID=NCD dbms_type=ORA dbs_ora_tnsname=NCD_DVEBMGS00 dbs_ora_schema=SAPSR3 ORACLE_SID=NCD TNS_ADMIN=/usr/sap/NCD/SYS/profile/oracle ORACLE_HOME=/oracle/NCD/19 NLS_LANG=AMERICAN_AMERICA.UTF8 SAPDATA_HOME=/oracle/NCD ORACLE_BASE=/oracle/NCD

SI/RAC	SI/RAC	Check SQL*Net connectivity to the new PDB for all relevant SAP application servers and database services.
		• On the database servers:
		`CMD> sqlplus sapsr3/'<password>'@NCD_DVEBMGS00`
		`CMD> sqlplus sapsr3/'<password>'@NCD_DV01`
		• On the application servers as `<sid>adm`:
		`CMD> R3trans -d`

SI/RAC	SI/RAC	Create the SAP-specific users and roles in the PDB by running the following PL/SQL scripts in the CDB. The scripts are shipped with BR*Tools. Always use the latest version of BR*Tools.
		`CMD> sqlplus /nolog @sapdba_role NCD`
		`CMD> sqlplus /nolog @sapconn_role NCD`
		`CMD> sqlplus /nolog @sapuprof_profile NCD`

Migrating a Database by Using Oracle RMAN

Each database migration that relies solely on Oracle RMAN is performed in two major steps:

1. Install a new system, as described in the referenced documents for installing SAP NetWeaver-based systems on Exadata Cloud Service or Exadata Cloud@Customer.
2. Replace the SAP database in the new installation with the database that you are migrating.

The advantage of this approach is that most SAP-relevant software installations and configurations of the Exadata compute nodes are performed by SAP SWPM before the initial database is replaced by the one that is being migrated.

For all RMAN-based migrations, it is mandatory to apply the same SAP Bundle Patch on the source system and the target system to avoid issues during restore or recovery, or conflicts that might arise because of minor incompatibilities.

The examples in the “Back Up, Restore, and Recover Method” and “Duplicate Database Method” sections assume that migration is performed from and to only one database instance and that any necessary modifications (for example, for Oracle RAC) are part of the post-migration work. Even if the source system and target system have the same number of database compute nodes, migrations using RMAN might require several database-related post-migration steps. Those steps are discussed in the “Post-Migration Tasks” section.

Prerequisites

Consider the following prerequisites before you begin the migration.

Oracle Database Home Software Versions and Patches

If the source database is version 19c, you must apply the latest SAP Bundle Patch on the source and target systems. If your source database is still unencrypted, the SAP Bundle Patch contains important patches to encrypt tablespaces as needed when the database is being restored. This requires a TDE encryption wallet set up on the source database.

If the source database is version 12.1, 12.2, or 18c, it cannot be encrypted when the database is being restored. Instead, the database is restored and recovered from the backup and then encrypted in a second step.

In either case, software versions and patches in both the source and target Oracle Database home must be the same.

For more information about restrictions imposed by SAP Bundle Patch levels on the database migration process, see SAP Note 3110260 - Oracle: SAP Bundle Patches and Oracle Database Migration.

TDE Setup

The best time to set up and implement TDE is *before* you migrate the database. This lets you back up the encrypted database on the source, restore and recover the database on the target, and keep existing encryption keys. In this case, the database is backed up on the source and restored on the target (possibly to a different location), and the encryption keys are transported from the encryption wallet on the source to a new encryption wallet on the target.

The next best option for databases already using version 19c is to set up a TDE encryption wallet on the source without encrypting any tablespaces. This allows TDE encryption of all tablespaces during database restore on the target.

If these options are not feasible, you can introduce TDE encryption only on the target by setting up a new encryption wallet with a new master key. The database is restored unencrypted and must be encrypted manually after recovery.

Back Up, Restore, and Recover Method

This section provides the major steps and several examples for using the RMAN Back Up, Restore, and Recover Database method for migration.

Prerequisites

This migration method is based on backing up the database on a source system and restoring it on a target system. So, you should choose a backup destination that the source and target can access or that you can detach from the source after backup and attach to the target before restore.

The easiest method is to mount a shared NFS file system on the source and target that is large enough to keep a full backup of the database. You could use an Exadata ASM Cluster File System exported to source or an external ZFS appliance.

Migration Steps

This section provides detailed steps for performing the migration. Detailed examples are provided in the sections that follow.

1. Create an online backup of the source database that includes the archive logs and controlfile.
2. Create an Oracle parameter file (`pfile`) that contains all the `init.ora` parameters of the source system.
3. If the source database is already encrypted with TDE, transport the contents of the encryption wallet to the target:
   A. Create a temporary encryption wallet on the file system.
   B. Merge the encryption keys from the source into the temporary encryption wallet.
   C. If required, copy the temporary encryption wallet to a file system location that can be accessed from the target system.
4. If the source database is not encrypted but TDE has been set up on it (it has an active encryption wallet and master key but no encrypted tablespaces), perform one of the following actions:
   - (Recommended) Set up a new encryption wallet with a master key on the target after you restore the controlfile from the backup**.**
   - Transport the encryption keys from the source to the target, as directed in step 3.
5. If the source database is not encrypted and TDE has *not* been set up (no active encryption wallet), you must create an encryption wallet with encryption keys and an autologin wallet on the target system in a later step.
6. Copy the Oracle parameter file (`pfile`) to the target system and make the parameter adjustments (for example, `control_files`, `audit_file_dest`, and `log_archive_dest_1`) required to start the target database instance into the `nomount` state.
7. Set up TDE configuration.
   - If the database is 19c, set up the `tde_configuration` and `wallet_root` initialization parameters as part of the TDE configuration.
   - If the database is 12.1, 12.2, or 18c, set up the TDE configuration in `sqlnet.ora`.
8. Start the target database instance into the `nomount` state by using the prepared `pfile`.
9. Restore the original controlfile and bring the database into the `mount` state.
10. Create the required directory for the encryption wallet in ASM.
11. If the source database was already encrypted with TDE, perform these steps:
    A. Create an empty encryption wallet on ASM on the target system.
    B. Merge the encryption keys from the previously created encryption wallet into the new encryption wallet on ASM.
    C. Create an autologin wallet.
12. If the source database was not encrypted but was configured for TDE, perform one of the following actions:
    - (Recommended) Create an encryption wallet with a master key.
    - Transport the encryption keys from the source into a new encryption wallet on the target, as directed in step 11.

13. If the source database was not encrypted and was *not* configured for TDE, perform these steps:
    A. Create an empty encryption wallet on ASM on the target system.
    B. Create an autologin wallet.
14. Restore and recover your database.
    - If the source database was already encrypted, restore and recover the database on the target.
    - If the source database was unencrypted but was on 19c with the latest SAP Bundle Patches and had TDE set up, use the `as encrypted` clause during database restore.
    - If the source database was on 12.1, 12.2, or 18c and was unencrypted, defer the encryption of the database until your database is successfully restored, recovered, and opened.
15. Open the database with the `resetlogs` option
16. If the source database was not encrypted and was *not* configured for TDE, create and set a new master key and restart the database instance.

    If the source database was unencrypted and could not be restored and encrypted during step 14, the next step is to encrypt it. You can perform this step online or offline.
    - The offline approach (see step 17) can be optimized to complete encryption much faster than the online approach.
    - The online approach (see step 18) lets you access the database much earlier while encryption is in progress.
17. To perform an offline encryption, follow these steps:
    A. Encrypt the SYSTEM tablespace and all the UNDO tablespaces offline (in the `mount` state).
    B. Open the database and set all the tablespaces *except* the SYSTEM and UNDO tablespaces offline.
    C. Encrypt each data file of those tablespaces. You can parallelize this task by scheduling the required DDL commands as jobs using the `dbms_scheduler` package.
    D. When all the data files are encrypted, set all the tablespaces back online.
    E. Create an encrypted temporary tablespace to replace the unencrypted PSAPTEMP tablespace.
    F. Change the default temporary tablespace to the new encrypted temporary tablespace, drop the old PSAPTEMP tablespace, and rename the new encrypted temporary tablespace to PSAPTEMP.
18. To perform an online encryption, perform one of the following steps:
    - Open the unencrypted database and use Oracle online encryption to encrypt all the tablespaces.

      This operation works on the tablespace level and cannot be parallelized on the data-file level (as it can in offline encryption). Online encryption can be done either serially (tablespace by tablespace) or in parallel (more than one tablespace at the same time), although each tablespace is encrypted by only one process. This means that the encryption of large tablespaces with many data files might take a long time. In addition, each unencrypted data file is encrypted into a new encrypted data file with a different name.
    - Open the unencrypted database, set up SAP BR*Tools, and use the `tablespace creation` and `tablespace reorganization` commands to create new encrypted tablespaces and to move your tables into the encrypted tablespaces.

Example 1: RMAN Back Up, Restore, and Recover from an Encrypted Oracle 19c Source Database

This example illustrates the RMAN Back Up, Restore, and Recover migration method in detail by migrating an encrypted source database, ODA, from an on-premises Exadata to Exadata Cloud@Customer. It uses an NFS file system mounted at `/backup` as a shared media for backups.

You configure the RMAN backup targets (for example, on NFS) especially for the migration, and create an online backup of the source database that includes archive logs, the controlfile, and optionally, the spfile. You then revert the backup targets to the old location after creating the backup.

On the source system, perform the following steps:

1. Note your current backup targets.

```
[oracle@myhost1 dbs]$ rman target /

Recovery Manager: Release 19.0.0.0.0 - Production on Tue Jan 5 15:30:32 2021
Version 19.9.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

connected to target database: ODA (DBID=3151516788)

RMAN> show all;

using target database control file instead of recovery catalog
RMAN configuration parameters for database with db_unique_name ODA are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP ON; # default
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '<orig_location>/%F';
CONFIGURE DEVICE TYPE DISK PARALLELISM 16 BACKUP TYPE TO BACKUPSET;
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT   '<orig_location>/%U';
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE OFF; # default
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
CONFIGURE COMPRESSION ALGORITHM 'BASIC' AS OF RELEASE 'DEFAULT' OPTIMIZE FOR LOAD TRUE ; #
default
CONFIGURE RMAN OUTPUT TO KEEP FOR 7 DAYS; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/oracle/ODA/19/dbs/snapcf_ODA.f'; # default
```

2. As the `root` user, run the following commands:

```
mkdir /backup/mig_oda
chown oracle:oinstall /backup/mig_oda
```

3. As the `oracle` user with the source database environment set, run the following commands:

```
[oracle@ dbs]$ rman target /

CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '/backup/mig_oda/%F';
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '/backup/mig_oda/%U' MAXPIECESIZE 128 G;
SET ENCRYPTION OFF;

BACKUP DATABASE CURRENT CONTFOLFILE SPFILE PLUS ARCHIVELOG;
```

4. Create an Oracle parameter file (`pfile`) that contains all the `init.ora` parameters of your source system.

```
SQL> create pfile=’/backup/mig_oda/init_oda.ora’ from spfile;
```

5. Restore the original backup locations.

```
[oracle@ dbs]$ rman target /

CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '<orig_location>/%F';
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '<orig_location>/%U' MAXPIECESIZE 128 G;
```

6. Prepare to copy the encryption keys from the encryption wallet of the source database.

   A. Create a temporary encryption wallet on the file system.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/tmp/' identified by "<password>";

keystore altered.
```

   B. Merge the encryption keys from the encryption wallet on the source into the temporary encryption wallet.

```
SQL> ADMINISTER KEY MANAGEMENT MERGE KEYSTORE '+DATAC1/ODA/orawallet/tde/' IDENTIFIED BY
"<password>" INTO EXISTING KEYSTORE '/tmp/' IDENTIFIED BY "<password>" WITH BACKUP;

keystore altered.
```

On the target system, perform the following steps:

1. Make necessary adjustments for the target environment in the parameter file by adjusting file system and ASM disk group locations (for example, `audit_file_dest` and `db_recovery_file_dest`) and by adding the initialization parameters for TDE. Then, copy the parameter file to the `dbs` directory under the Oracle Database home (`ORACLE_HOME`) on your target system. The parameter file needs to be only on one cluster node because it is required only during the migration of the database.

```
tde_configuration = ‘KEYSTORE_CONFIGURATION=FILE’
wallet_root = ‘+DATAC2/ODA/orawallet’
```

2. Create the required wallet locations on ASM.

```
[oracle@node1 ~]$ asmcmd
ASMCMD> cd +DATAC2
ASMCMD> mkdir ODA
ASMCMD> cd ODA
ASMCMD> mkdir orawallet
ASMCMD> mkdir orawallet/tde
```

3. Start the target database instance with the prepared parameter file into the `nomount` state.

```
SQL> startup nomount pfile='?/dbs/init_oda.ora';
ORACLE instance started.

Total System Global Area 6.8719E+10 bytes
Fixed Size                 26611816 bytes
Variable Size            3.3018E+10 bytes
Database Buffers         3.5568E+10 bytes
Redo Buffers              107601920 bytes
```

4. Create an encryption wallet on ASM, merge the encryption keys from the temporary encryption wallet, and shut down the database instance.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE identified by "<password>";

keystore altered.
```

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "<password>";

keystore altered.

SQL> ADMINISTER KEY MANAGEMENT MERGE KEYSTORE '/tmp/' IDENTIFIED BY "<password>" INTO EXISTING
KEYSTORE '+DATAC1/ODA/orawallet/tde/' IDENTIFIED BY "<password>" WITH BACKUP;

keystore altered.

SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
```

5. Once more, start the target database instance with the prepared parameter file into the `nomount` state.

```
SQL> startup nomount pfile='?/dbs/init_oda.ora';
ORACLE instance started.

Total System Global Area 6.8719E+10 bytes
Fixed Size                 26611816 bytes
Variable Size            3.3018E+10 bytes
Database Buffers         3.5568E+10 bytes
Redo Buffers              107601920 bytes
```

6. Open the new encryption wallet and create an autologin wallet so that it opens automatically when the database instance is started.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "<password>";

keystore altered.

SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE
'+DATAC1/ODA/orawallet/tde/' IDENTIFIED BY "<password>";

keystore altered.
```

7. Once more, shut down and then start the target database instance with the prepared parameter file into the `nomount` state.

```
SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.

SQL> startup nomount pfile='?/dbs/init_oda.ora';
ORACLE instance started.

Total System Global Area 6.8719E+10 bytes
Fixed Size                 26611816 bytes
Variable Size            3.3018E+10 bytes
Database Buffers         3.5568E+10 bytes
Redo Buffers              107601920 bytes
```

8. Verify that the wallet opens and that the master key is present.

```
SQL> select * from v$encryption_wallet;

WRL_TYPE
--------------------
```

```
WRL_PARAMETER
--------------------------------------------------------------------------------
STATUS                         WALLET_TYPE          WALLET_OR KEYSTORE FULLY_BAC
------------------------------ -------------------- --------- -------- ---------
    CON_ID
----------
ASM
+DATAC1/ODA/orawallet/tde/
OPEN                           AUTOLOGIN            SINGLE    NONE     NO
         0


SQL> select max(key_id) from v$encryption_keys;

MAX(KEY_ID)
--------------------------------------------------------------------------------
Ae/pik4KoU/ov/VbX9QoMEkAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

9. Restore the controlfile from backup.

```
RMAN> restore controlfile from '/backup/mig_oda/c-3151516788-20201009-00';

Starting restore at 09-OCT-20
using target database controlfile instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=2109 device type=DISK

channel ORA_DISK_1: restoring control file
channel ORA_DISK_1: restore complete, elapsed time: 00:00:03
output file name=+DATAC1/ODA/cntrloda.dbf
output file name=+RECOC1/ODA/cntrloda.dbf
Finished restore at 09-OCT-20
```

10. Mount the database.

```
RMAN> alter database mount;
```

11. Create the list of the `set newname` commands to be used in the next step. For example:

```
set linesize 1000
set serveroutput on
spool setnewname.sql

declare
  src_pat varchar2(50):='+DATAC1';
  tgt_pat varchar2(50):='+DATAC2';
  old_name varchar2(2000);
  new_name varchar2(2000);
begin
  src_pat:='^'||src_pat;
  for c1 in (select name from v$datafile) loop
    old_name := ''''||c1.name||'''';
    new_name := ''''||REGEXP_REPLACE(c1.name,src_pat,tgt_pat)||'''';
    dbms_output.put_line('set newname for datafile '||old_name||' to '||new_name||';');
  end loop;
  for c1 in (select name from v$tempfile) loop
    old_name := ''''||c1.name||'''';
    new_name := ''''||REGEXP_REPLACE(c1.name,src_pat,tgt_pat)||'''';
    dbms_output.put_line('set newname for tempfile '||old_name||' to '||new_name||';');
```

```
  end loop;
end;
/
spool off
```

12. If necessary, adjust the locations and names of all database-related files by using the `set newname` command during restore.

```
RMAN> catalog start with '/backup/mig_oda/';
run {
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/system.295.1042756999' to
'+DATAC2/ODA/DATAFILE/system.295.1042756999';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/sysaux.296.1042756999' to
'+DATAC2/ODA/DATAFILE/sysaux.296.1042756999';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapundo001.297.1042757001' to
'+DATAC2/ODA/DATAFILE/psapundo001.297.1042757001';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3.300.1042757673' to
'+DATAC2/ODA/DATAFILE/psapsr3.300.1042757673';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3.301.1042757675' to
'+DATAC2/ODA/DATAFILE/psapsr3.301.1042757675';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3.302.1042757677' to
'+DATAC2/ODA/DATAFILE/psapsr3.302.1042757677';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3.303.1042757677' to
'+DATAC2/ODA/DATAFILE/psapsr3.303.1042757677';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3.304.1042757679' to
'+DATAC2/ODA/DATAFILE/psapsr3.304.1042757679';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3.305.1042757681' to
'+DATAC2/ODA/DATAFILE/psapsr3.305.1042757681';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3750.306.1042757681' to
'+DATAC2/ODA/DATAFILE/psapsr3750.306.1042757681';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3750.307.1042757683' to
'+DATAC2/ODA/DATAFILE/psapsr3750.307.1042757683';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3750.308.1042757685' to
'+DATAC2/ODA/DATAFILE/psapsr3750.308.1042757685';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3750.309.1042757687' to
'+DATAC2/ODA/DATAFILE/psapsr3750.309.1042757687';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3750.310.1042757687' to
'+DATAC2/ODA/DATAFILE/psapsr3750.310.1042757687';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3usr.311.1042757689' to
'+DATAC2/ODA/DATAFILE/psapsr3usr.311.1042757689';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapundo002.312.1042757689' to
'+DATAC2/ODA/DATAFILE/psapundo002.312.1042757689';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/big.341.1049019829' to
'+DATAC2/ODA/DATAFILE/big.341.1049019829';

set NEWNAME FOR TEMPFILE '+DATAC1/ODA/TEMPFILE/psaptemp.298.1042757001' to
'+DATAC2/ODA/TEMPFILE/psaptemp.298.1042757001';

restore database;
switch datafile all;
switch tempfile all;
}
```

13. Recover and open the database.

```
RMAN> recover database;

RMAN> alter database open resetlogs;
```

Example 2: RMAN Back Up, Restore, and Recover from an Unencrypted Oracle 19c Source Database with Existing TDE Configuration

This example illustrates the RMAN Back Up, Restore, and Recover migration method in detail by migrating an unencrypted source database, ODA, that has TDE configuration (an encryption wallet) from an on-premises Exadata to Exadata Cloud@Customer. It uses an NFS file system mounted at `/backup` as a shared media for backups. The database is encrypted while it is being restored.

You configure the RMAN backup targets (for example, on NFS) especially for the migration, and create an online backup of the source database that includes archive logs, the controlfile, and optionally, the spfile. Then you revert the backup targets to the old location after creating the backup.

On the source system, perform the following steps:

1. Create the required wallet location on ASM.

```
[oracle@node1 ~]$ asmcmd
ASMCMD> cd +DATAC1
ASMCMD> mkdir ODA
ASMCMD> cd ODA
ASMCMD> mkdir orawallet
ASMCMD> mkdir orawallet/tde
```

Note: If you are not using ASM on the source, create the directory structure for the wallet on a file system.

2. Make necessary adjustments in the server parameter file on the source by adding the initialization parameters for TDE.

```
tde_configuration = 'KEYSTORE_CONFIGURATION=FILE'
wallet_root = '+DATAC1/ODA/orawallet'
```

3. Restart the database instances to make the changes take effect.

4. Create the encryption wallet and autologin wallet.

```
SQL> select key_id,KEY_USE from v$encryption_keys;

no rows selected

SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE IDENTIFIED BY "<password>";

keystore altered.

SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE
'+DATAC1/ODA/orawallet/tde/' IDENTIFIED BY "<password>";

keystore altered.

SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "<password>";

keystore altered.

SQL> alter system set "_db_discard_lost_masterkey"=TRUE;

System altered.
```

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY USING TAG 'MASTERKEY' FORCE KEYSTORE IDENTIFIED BY
"<password>" WITH BACKUP;

keystore altered.
```

The TDE configuration on the source is complete.

5. Note the current backup targets.

```
[oracle@myhost1 dbs]$ rman target /

Recovery Manager: Release 19.0.0.0.0 - Production on Tue Jan 5 15:30:32 2021
Version 19.9.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

connected to target database: ODA (DBID=3151516788)

RMAN> show all;

using target database control file instead of recovery catalog
RMAN configuration parameters for database with db_unique_name ODA are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP ON; # default
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '<orig_location>/%F';
CONFIGURE DEVICE TYPE DISK PARALLELISM 16 BACKUP TYPE TO BACKUPSET;
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT   '<orig_location>/%U';
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE OFF; # default
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
CONFIGURE COMPRESSION ALGORITHM 'BASIC' AS OF RELEASE 'DEFAULT' OPTIMIZE FOR LOAD TRUE ; #
default
CONFIGURE RMAN OUTPUT TO KEEP FOR 7 DAYS; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/oracle/ODA/19/dbs/snapcf_ODA.f'; # default
```

6. As the `root` user, run the following commands:

```
mkdir /backup/mig_oda
chown oracle:oinstall /backup/mig_oda
```

7. As the `oracle` user with the source database environment set, run the following commands:

```
[oracle@ dbs]$ rman target /

CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '/backup/mig_oda/%F';
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '/backup/mig_oda/%U' MAXPIECESIZE 128 G;
SET ENCRYPTION OFF;

BACKUP DATABASE CURRENT CONTFOLFILE SPFILE PLUS ARCHIVELOG;
```

8. Create an Oracle parameter file (`pfile`) that contains all the `init.ora` parameters of the source system.

```
SQL> create pfile=’/backup/mig_oda/init_oda.ora’ from spfile;
```

9. Restore original backup locations.

```
[oracle@ dbs]$ rman target /

CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '<orig_location>/%F';
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '<orig_location>/%U' MAXPIECESIZE 128 G;
```

On the target system, perform the following steps:

1. Make necessary adjustments for the target environment in the parameter file by adjusting file system and ASM disk group locations (for example, `audit_file_dest` and `db_recovery_file_dest`) and by adding the initialization parameters for TDE. Then, copy the parameter file to the `dbs` directory under the Oracle Database home (`ORACLE_HOME`) on your target system. The parameter file needs to be only on one cluster node because it is required only during the migration of the database.

```
tde_configuration = 'KEYSTORE_CONFIGURATION=FILE'
wallet_root = '+DATAC2/ODA/orawallet'
```

2. Create the required wallet locations on ASM.

```
[oracle@node1 ~]$ asmcmd
ASMCMD> cd +DATAC2
ASMCMD> mkdir ODA
ASMCMD> cd ODA
ASMCMD> mkdir orawallet
ASMCMD> mkdir orawallet/tde
```

3. Start the target database instance with the prepared parameter file into the `nomount` state.

```
SQL> startup nomount pfile='?/dbs/init_oda.ora';
ORACLE instance started.

Total System Global Area 6.8719E+10 bytes
Fixed Size                 26611816 bytes
Variable Size            3.3018E+10 bytes
Database Buffers         3.5568E+10 bytes
Redo Buffers              107601920 bytes
```

4. Restore the controlfile from backup.

```
RMAN> restore controlfile from '/backup/mig_oda/c-3151516788-20201009-00';

Starting restore at 09-OCT-20
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=2109 device type=DISK

channel ORA_DISK_1: restoring control file
channel ORA_DISK_1: restore complete, elapsed time: 00:00:03
output file name=+DATAC1/ODA/cntrloda.dbf
output file name=+RECOC1/ODA/cntrloda.dbf
Finished restore at 09-OCT-20
```

5. Mount the database.

```
RMAN> alter database mount;
```

6. Create the list of the `set newname` commands to be used in the next step. For example:

```
set linesize 1000
set serveroutput on
spool setnewname.sql
```

```
declare
  src_pat varchar2(50):='+DATAC1';
  tgt_pat varchar2(50):='+DATAC2';
  old_name varchar2(2000);
  new_name varchar2(2000);
begin
  src_pat:='^'||src_pat;
  for c1 in (select name from v$datafile) loop
    old_name := ''''||c1.name||'''';
    new_name := ''''||REGEXP_REPLACE(c1.name,src_pat,tgt_pat)||'''';
    dbms_output.put_line('set newname for datafile '||old_name||' to '||new_name||';');
  end loop;
  for c1 in (select name from v$tempfile) loop
    old_name := ''''||c1.name||'''';
    new_name := ''''||REGEXP_REPLACE(c1.name,src_pat,tgt_pat)||'''';
    dbms_output.put_line('set newname for tempfile '||old_name||' to '||new_name||';');
  end loop;
end;
/
spool off
```

7. If necessary, adjust the locations and names of all database-related files by using the `set newname` command during restore with the `as encrypted` clause.

```
RMAN> catalog start with '/backup/mig_oda/';
run {
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/system.295.1042756999' to
'+DATAC2/ODA/DATAFILE/system.295.1042756999';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/sysaux.296.1042756999' to
'+DATAC2/ODA/DATAFILE/sysaux.296.1042756999';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapundo001.297.1042757001' to
'+DATAC2/ODA/DATAFILE/psapundo001.297.1042757001';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3.300.1042757673' to
'+DATAC2/ODA/DATAFILE/psapsr3.300.1042757673';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3.301.1042757675' to
'+DATAC2/ODA/DATAFILE/psapsr3.301.1042757675';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3.302.1042757677' to
'+DATAC2/ODA/DATAFILE/psapsr3.302.1042757677';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3.303.1042757677' to
'+DATAC2/ODA/DATAFILE/psapsr3.303.1042757677';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3.304.1042757679' to
'+DATAC2/ODA/DATAFILE/psapsr3.304.1042757679';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3.305.1042757681' to
'+DATAC2/ODA/DATAFILE/psapsr3.305.1042757681';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3750.306.1042757681' to
'+DATAC2/ODA/DATAFILE/psapsr3750.306.1042757681';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3750.307.1042757683' to
'+DATAC2/ODA/DATAFILE/psapsr3750.307.1042757683';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3750.308.1042757685' to
'+DATAC2/ODA/DATAFILE/psapsr3750.308.1042757685';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3750.309.1042757687' to
'+DATAC2/ODA/DATAFILE/psapsr3750.309.1042757687';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3750.310.1042757687' to
'+DATAC2/ODA/DATAFILE/psapsr3750.310.1042757687';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3usr.311.1042757689' to
'+DATAC2/ODA/DATAFILE/psapsr3usr.311.1042757689';
```

```
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapundo002.312.1042757689' to
'+DATAC2/ODA/DATAFILE/psapundo002.312.1042757689';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/big.341.1049019829' to
'+DATAC2/ODA/DATAFILE/big.341.1049019829';

set NEWNAME FOR TEMPFILE '+DATAC1/ODA/TEMPFILE/psaptemp.298.1042757001' to
'+DATAC2/ODA/TEMPFILE/psaptemp.298.1042757001';

restore database as encrypted;
switch datafile all;
switch tempfile all;
}
```

8. Recover and open the database.

```
RMAN> recover database;

RMAN> alter database open resetlogs;
```

All the tablespaces should be reported as encrypted.

```
SQL> select TS#,ENCRYPTIONALG,ENCRYPTEDTS,STATUS from v$encrypted_tablespaces;

       TS# ENCRYPT ENC STATUS
---------- ------- --- ----------
         0 AES128  YES NORMAL
         1 AES128  YES NORMAL
         2 AES128  YES NORMAL
         4 AES128  YES NORMAL
         5 AES128  YES NORMAL
         6 AES128  YES NORMAL
         7 AES128  YES NORMAL
         8 AES128  YES NORMAL
         9 AES128  YES NORMAL
```

Example 3: RMAN Back Up, Restore, and Recover from an Unencrypted Oracle 19c Source Database with No TDE Configuration

This example illustrates the RMAN Back Up, Restore, and Recover migration method in detail by migrating an unencrypted source database, ODA, without any TDE configuration from an on-premises Exadata to Exadata Cloud@Customer. It uses an NFS file system mounted at `/backup` as a shared media for backups. At the end of this example, two Oracle native encryption methods are shown: offline and online.

You configure the RMAN backup targets (for example, on NFS) especially for the migration, and create an online backup of the source database that includes archive logs, the controlfile, and optionally, the spfile. You then revert the backup targets to the old location after creating the backup.

On the source system, perform the following steps:

1. Note the current backup targets.

```
[oracle@myhost1 dbs]$ rman target /

Recovery Manager: Release 19.0.0.0.0 - Production on Tue Jan 5 15:30:32 2021
Version 19.9.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

connected to target database: ODA (DBID=3151516788)
```

```
RMAN> show all;

using target database control file instead of recovery catalog
RMAN configuration parameters for database with db_unique_name ODA are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP ON; # default
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '<orig_location>/%F';
CONFIGURE DEVICE TYPE DISK PARALLELISM 16 BACKUP TYPE TO BACKUPSET;
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT   '<orig_location>/%U';
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE OFF; # default
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
CONFIGURE COMPRESSION ALGORITHM 'BASIC' AS OF RELEASE 'DEFAULT' OPTIMIZE FOR LOAD TRUE ; #
default
CONFIGURE RMAN OUTPUT TO KEEP FOR 7 DAYS; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/oracle/ODA/19/dbs/snapcf_ODA.f'; # default
```

2. As the `root` user, run the following commands:

```
mkdir /backup/mig_oda
chown oracle:oinstall /backup/mig_oda
```

3. As the `oracle` user with the source database environment set, run the following commands:

```
[oracle@ dbs]$ rman target /

CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '/backup/mig_oda/%F';
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '/backup/mig_oda/%U' MAXPIECESIZE 128 G;
SET ENCRYPTION OFF;

BACKUP DATABASE CURRENT CONTFOLFILE SPFILE PLUS ARCHIVELOG;
```

4. Create an Oracle parameter file (`pfile`) that contains all the `init.ora` parameters of your source system.

```
SQL> create pfile=’/backup/mig_oda/init_oda.ora’ from spfile;
```

5. Restore the original backup locations.

```
[oracle@ dbs]$ rman target /

CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '<orig_location>/%F';
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '<orig_location>/%U' MAXPIECESIZE 128 G;
```

On the target system, perform the following steps:

1. Make necessary adjustments for the target environment in the parameter file by adjusting file system and ASM disk group locations (for example, `audit_file_dest` and `db_recovery_file_dest`) and by adding the initialization parameters for TDE. Then, copy the parameter file to the `dbs` directory under the Oracle Database home (`ORACLE_HOME`) on your target system. The parameter file needs to be only on one cluster node because it is required only during the migration of the database.

```
tde_configuration = ‘KEYSTORE_CONFIGURATION=FILE’
wallet_root = ‘+DATAC2/ODA/orawallet’
```

2. Create the required wallet locations on ASM.

```
[oracle@node1 ~]$ asmcmd
ASMCMD> cd +DATAC2
ASMCMD> mkdir ODA
ASMCMD> cd ODA
ASMCMD> mkdir orawallet
ASMCMD> mkdir orawallet/tde
```

3. Start the target database instance with the prepared parameter file into the `nomount` state.

```
SQL> startup nomount pfile='?/dbs/init_oda.ora';
ORACLE instance started.

Total System Global Area 6.8719E+10 bytes
Fixed Size                 26611816 bytes
Variable Size            3.3018E+10 bytes
Database Buffers         3.5568E+10 bytes
Redo Buffers              107601920 bytes
```

4. Restore the controlfile from backup.

```
RMAN> restore controlfile from '/backup/mig_oda/c-3151516788-20201009-00';

Starting restore at 09-OCT-20
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=2109 device type=DISK

channel ORA_DISK_1: restoring control file
channel ORA_DISK_1: restore complete, elapsed time: 00:00:03
output file name=+DATAC1/ODA/cntrloda.dbf
output file name=+RECOC1/ODA/cntrloda.dbf
Finished restore at 09-OCT-20
```

5. Mount the database.

```
RMAN> alter database mount;
```

6. Create the list of the `set newname` commands to be used in the next step. For example:

```
set linesize 1000
set serveroutput on
spool setnewname.sql

declare
  src_pat varchar2(50):='+DATAC1';
  tgt_pat varchar2(50):='+DATAC2';
  old_name varchar2(2000);
  new_name varchar2(2000);
begin
  src_pat:='^'||src_pat;
  for c1 in (select name from v$datafile) loop
    old_name := ''''||c1.name||'''';
    new_name := ''''||REGEXP_REPLACE(c1.name,src_pat,tgt_pat)||'''';
    dbms_output.put_line('set newname for datafile '||old_name||' to '||new_name||';');
  end loop;
  for c1 in (select name from v$tempfile) loop
    old_name := ''''||c1.name||'''';
    new_name := ''''||REGEXP_REPLACE(c1.name,src_pat,tgt_pat)||'''';
    dbms_output.put_line('set newname for tempfile '||old_name||' to '||new_name||';');
```

```
  end loop;
end;
/
spool off
```

7. If necessary, adjust the locations and names of all database-related files by using the `set newname` command during restore.

```
RMAN> catalog start with '/backup/mig_oda/';
run {
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/system.295.1042756999' to
'+DATAC2/ODA/DATAFILE/system.295.1042756999';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/sysaux.296.1042756999' to
'+DATAC2/ODA/DATAFILE/sysaux.296.1042756999';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapundo001.297.1042757001' to
'+DATAC2/ODA/DATAFILE/psapundo001.297.1042757001';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3.300.1042757673' to
'+DATAC2/ODA/DATAFILE/psapsr3.300.1042757673';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3.301.1042757675' to
'+DATAC2/ODA/DATAFILE/psapsr3.301.1042757675';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3.302.1042757677' to
'+DATAC2/ODA/DATAFILE/psapsr3.302.1042757677';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3.303.1042757677' to
'+DATAC2/ODA/DATAFILE/psapsr3.303.1042757677';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3.304.1042757679' to
'+DATAC2/ODA/DATAFILE/psapsr3.304.1042757679';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3.305.1042757681' to
'+DATAC2/ODA/DATAFILE/psapsr3.305.1042757681';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3750.306.1042757681' to
'+DATAC2/ODA/DATAFILE/psapsr3750.306.1042757681';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3750.307.1042757683' to
'+DATAC2/ODA/DATAFILE/psapsr3750.307.1042757683';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3750.308.1042757685' to
'+DATAC2/ODA/DATAFILE/psapsr3750.308.1042757685';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3750.309.1042757687' to
'+DATAC2/ODA/DATAFILE/psapsr3750.309.1042757687';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3750.310.1042757687' to
'+DATAC2/ODA/DATAFILE/psapsr3750.310.1042757687';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapsr3usr.311.1042757689' to
'+DATAC2/ODA/DATAFILE/psapsr3usr.311.1042757689';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/psapundo002.312.1042757689' to
'+DATAC2/ODA/DATAFILE/psapundo002.312.1042757689';
set NEWNAME FOR DATAFILE '+DATAC1/ODA/DATAFILE/big.341.1049019829' to
'+DATAC2/ODA/DATAFILE/big.341.1049019829';

set NEWNAME FOR TEMPFILE '+DATAC1/ODA/TEMPFILE/psaptemp.298.1042757001' to
'+DATAC2/ODA/TEMPFILE/psaptemp.298.1042757001';

restore database;
switch datafile all;
switch tempfile all;
}
```

8. Recover and open the database.

```
RMAN> recover database;

RMAN> alter database open resetlogs;
```

9. Create an encryption wallet and autologin wallet.

```
SQL> select key_id,KEY_USE from v$encryption_keys;

no rows selected

SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE IDENTIFIED BY "<password>";

keystore altered.

SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE
'+DATAC1/ODA/orawallet/tde/' IDENTIFIED BY "<password>";

keystore altered.

SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "<password>";

keystore altered.

SQL> alter system set "_db_discard_lost_masterkey"=TRUE;

System altered.

SQL> ADMINISTER KEY MANAGEMENT SET KEY USING TAG 'MASTERKEY' FORCE KEYSTORE IDENTIFIED BY
"<password>" WITH BACKUP;

keystore altered.
```

10. For offline encryption of the whole database, follow these steps. If you want to perform online encryption, skip to step 11.

    A. Encrypt the SYSTEM and UNDO tablespaces in the `mount` state.

    B. Open the database.

    C. Set all the remaining permanent tablespaces offline.

    D. Start an encryption job for each data file.

> **Note**: The following example script uses all available Oracle job queue processes to encrypt data files as fast as possible in parallel and might consume a large amount of CPU resources. If other systems are running on the same host, we recommend configuring the Oracle job queue processes to a value that does not allocate more CPU resources than is reasonable.

```
shutdown immediate;
startup mount;

ALTER TABLESPACE SYSTEM ENCRYPTION OFFLINE ENCRYPT;
ALTER TABLESPACE PSAPUNDO001 ENCRYPTION OFFLINE ENCRYPT;
ALTER TABLESPACE PSAPUNDO002 ENCRYPTION OFFLINE ENCRYPT;
ALTER DATABASE OPEN;

declare
  cmd1 varchar2(32767);
  cmd2 varchar2(32767);
  job_cmd varchar2(32767);
  jobname varchar2(32767);
```

```
  prefix varchar2(512);
begin
  for t1 in (SELECT tablespace_name from dba_tablespaces where contents<>'TEMPORARY' and
contents<>'UNDO' and tablespace_name<>'SYSTEM') loop
    cmd1:='ALTER TABLESPACE '|| t1.tablespace_name ||' OFFLINE';
    begin
      execute immediate cmd1;
      exception
        when others then null;
    end;

    for f1 in (select file_name from dba_data_files where tablespace_name =
t1.tablespace_name) loop
      cmd2:='ALTER DATABASE DATAFILE '||''''||''''||f1.file_name||''''||''''||' ENCRYPT';

      job_cmd:='BEGIN EXECUTE IMMEDIATE '''||cmd2||'''; END;';

      prefix:=substr(t1.tablespace_name||'ENC',1,18);
      jobname:=dbms_scheduler.generate_job_name(prefix);
      dbms_scheduler.create_job(job_name => jobname,job_type => 'PLSQL_BLOCK',job_action =>
job_cmd,enabled => FALSE);
      dbms_scheduler.run_job(job_name => jobname,use_current_session => FALSE);
---      dbms_output.put_line(job_cmd);
    end loop;
  end loop;
end;
/
```

E. Regularly check the status of the encryption process. You can monitor the process in the following ways:

- Use the OS utility `top`, which shows numerous job queue processes consuming CPU time for encryption
- Use `V$ENCRYPTED_TABLESPACES`
- Check the status of the generated jobs in `DBA_SCHEDULER_JOBS`

For example:

```
Output of top:
376750 oracle    20   0   26.4g  99580  89440 S  26.8  0.1   0:02.93 ora_j007_MFG001
376512 oracle    20   0   26.7g  99684  89536 S  25.8  0.1   0:03.01 ora_j004_MFG001
376488 oracle    20   0   26.7g 106832  95292 S  24.8  0.1   0:03.12 ora_j000_MFG001
378349 oracle    20   0   26.7g  99120  89060 R  24.8  0.1   0:02.78 ora_j008_MFG001
378357 oracle    20   0   26.7g  99588  89432 R  24.5  0.1   0:02.46 ora_j00b_MFG001
376499 oracle    20   0   26.4g  98892  88844 R  22.5  0.1   0:02.85 ora_j003_MFG001
376495 oracle    20   0   26.7g  99832  89664 S  21.9  0.1   0:02.93 ora_j002_MFG001
376490 oracle    20   0   26.7g  99612  89472 S  21.2  0.1   0:02.58 ora_j001_MFG001
378360 oracle    20   0   26.4g 102676  92488 S  20.3  0.1   0:02.53 ora_j00c_MFG001
378353 oracle    20   0   26.7g  99784  89632 S  19.9  0.1   0:02.27 ora_j009_MFG001
376642 oracle    20   0   26.7g  99244  89164 S  19.6  0.1   0:02.82 ora_j006_MFG001
376558 oracle    20   0   26.4g  99284  89132 S  18.0  0.1   0:02.46 ora_j005_MFG001

Out of of V$ENCRYPTED_TABLESPACES:

SQL> select TS#,STATUS,ENCRYPTIONALG from V$ENCRYPTED_TABLESPACES;

       TS# STATUS     ENCRYPT
```

```
---------- ---------- -------
         0 NORMAL     AES128
         1 NORMAL     AES128
         2 NORMAL     AES128
         4 ENCRYPTING AES128
         5 ENCRYPTING AES128
         6 NORMAL     AES128
         7 NORMAL     AES128
         8 NORMAL     AES128
```

F. When all the data files are encrypted, set all the tablespaces back online. For example:

```
declare
  cmd1 varchar2(32767);
begin
  for t1 in (SELECT tablespace_name from dba_tablespaces where contents='PERMANENT' and
tablespace_name<>'SYSTEM') loop
    cmd1:='ALTER TABLESPACE '|| t1.tablespace_name ||' ONLINE';
    begin
      execute immediate cmd1;
      exception
        when others then null;
    end;
  end loop;
end;
/
```

11. For online encryption of the whole database, run the following PL/SQL script. Modify it to match your needs before running it.

Note: The following example script uses all available Oracle job queue processes to encrypt tablespaces as fast as possible in parallel. Depending on the number of tablespaces, it might consume a large amount of CPU resources. We recommend configuring the Oracle job queue processes to a value that will not allocate more CPU resources than is reasonable.

```
declare
  old_file varchar2(500);
  new_file varchar2(500);
  cmd varchar2(32767);
  job_cmd varchar2(32767);
  jobname varchar2(32767);
  file_nr number;
  suffix varchar2(16):='_enc';
  prefix varchar2(512);
begin
  for t1 in (select tablespace_name from dba_tablespaces where contents<>'TEMPORARY') loop
    file_nr := 1;
    cmd := 'ALTER TABLESPACE '||t1.tablespace_name||' ENCRYPTION ONLINE ENCRYPT
FILE_NAME_CONVERT=(';
    for f1 in (select file_name from dba_data_files where tablespace_name = t1.tablespace_name)
loop
      if file_nr>1 then
        cmd:=cmd||',';
      end if;
      old_file := f1.file_name;
      if instr(old_file,'.')>1 then
```

```
        new_file := substr(old_file,1,instr(old_file,'.')-
1)||lpad(to_char(file_nr),3,'0')||suffix||'.dbf';
      else
        new_file := old_file||lpad(to_char(file_nr),3,'0')||suffix||'.dbf';
      end if;
      file_nr:=file_nr+1;
      cmd:=cmd||''''||''''||old_file||''''||''''||',||''''||''''||new_file||''''||'''';
    end loop;
    cmd:=cmd||')';

    job_cmd:='BEGIN EXECUTE IMMEDIATE '''||cmd||'''; END;';

    prefix:=substr(t1.tablespace_name||'ENC',1,18);
    jobname:=dbms_scheduler.generate_job_name(prefix);
    dbms_scheduler.create_job(job_name => jobname,job_type => 'PLSQL_BLOCK',job_action =>
job_cmd,enabled => FALSE);
    dbms_scheduler.run_job(job_name => jobname,use_current_session => FALSE);
  end loop;
end;
/
```

12. Create an encrypted temporary tablespace to replace your unencrypted PSAPTEMP tablespace.

```
CREATE TEMPORARY TABLESPACE PSAPTEMP_ENC TEMPFILE '+DATAC2' SIZE 32767M ENCRYPTION ENCRYPT;
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE PSAPTEMP_ENC;
DROP TABLESPACE PSAPTEMP;
ALTER TABLESPACE PSAPTEMP_ENC RENAME TO PSAPTEMP;
```

Duplicate Database Method

This section provides the major steps and several examples for using the RMAN Duplicate Database method for migration.

Prerequisites

RMAN Duplicate Database does not require a shared file system for backup and restore. Instead, it uses a SQL*Net connection to copy the mounted or opened source database to the target. If the copy was taken from a source database in the open state, the target database is automatically recovered and opened at the end of the process.

Because this method uses SQL*Net, we strongly recommend configuring dedicated SQL*Net connections and Oracle listeners solely for the duplication process. Doing so avoids conflicts with existing Oracle listeners controlled by Oracle Grid Infrastructure, and the listeners can be removed quickly after duplication. Also, SQL*Net and listeners are configured to work in two directions (source-to-target and target-to-source) because RMAN can either push or pull the source database. This depends on where it is started, the source or target system.

The RMAN `as encrypted` clause is supported only with Oracle Database 19c and the latest SAP Bundle Patch. If your source database is not already encrypted with TDE, then encryption is a post-duplication task for Oracle Database 12.1, 12.2, and 18c.

Migration Steps

This section provides detailed steps for performing the migration. Detailed examples are provided in the sections that follow.

1. Create a password file on the target under `$ORACLE_HOME/dbs`.
2. Prepare an Oracle initialization parameter file (`pfile`) to start the target database instance into the `nomount` state.
   - If the database is 19c, set up the `tde_configuration` and `wallet_root` initialization parameters as part of the TDE configuration.
   - If the database is 12.1, 12.2, or 18c, set up your TDE configuration in `sqlnet.ora`.
   - Set up initialization parameters (for example, `db_file_name_convert` or `log_file_name_convert`) for name conversions if necessary or, alternatively, use the `set newname` option during duplication.
3. Configure SQL*Net and Oracle listeners on the source and target, and verify that the connections work in both directions.
4. If the source database is already encrypted with TDE, transport the contents of the encryption wallet to the target:
   A. Create a temporary encryption wallet on the file system.
   B. Merge the encryption keys from the source into the temporary encryption wallet.
   C. If required, copy the temporary encryption wallet to a file system location that can be accessed from the target system.
5. If the source database is not encrypted but TDE has been set up on it (it has an active encryption wallet and master key but no encrypted tablespaces), perform one of the following actions:
   - (Recommended) Set up a new encryption wallet with a master key on the target after you restore the controlfile from the backup**.**
   - Migrate the encryption keys from the source to the target, as directed in step 4.
6. If the source database is not encrypted and TDE has *not* been set up (no active encryption wallet), you must create an encryption wallet with encryption keys and an autologin wallet on the target system in a later step.
7. Place your prepared Oracle parameter file (`pfile`) on the target system and start the target database instance into the `nomount` state.
8. Create the required directory for the encryption wallet in ASM.
9. If the source database was encrypted with TDE, perform these steps:
   A. Create an empty encryption wallet on ASM on the target system.
   B. Merge the encryption keys from the previously created temporary encryption wallet into the new encryption wallet on ASM and open it.
   C. Create an autologin wallet.

10. If the source database was not encrypted but was configured for TDE, perform one of the following actions:
    - Create an encryption wallet with a master key *after duplication* of the database and encrypt your database manually.
    - If you want to run RMAN duplicate database with the `as encrypted` clause, transport the encryption keys from the source into a new encryption wallet on the target, as directed in step 9. Shut down the database instance and start it again into the `nomount` state. Verify that the encryption wallet opens automatically and that the encryption keys are present.
11. If the source database was not encrypted and was *not* configured for TDE, create an encryption wallet with a master key *after duplication* of the database and encrypt the database manually.
12. Run duplication of the source database.
13. If you decided to run RMAN duplicate database without the `as encrypted` clause, create an encryption wallet with a new master key and encrypt your database manually by performing the following steps:
    A. Create an empty encryption wallet on ASM on the target system.
    B. Create and set a master key in the new encryption wallet.
    C. Create an autologin wallet.
    D. Encrypt the database manually.

Example 1: RMAN Duplicate from an Encrypted Oracle 19c Source Database

This example illustrates the RMAN Duplicate Database migration method in detail by migrating an encrypted source database, ODA, from an on-premises Exadata to Exadata Cloud@Customer.

On the source system, perform the following steps:

1. Create a temporary encryption wallet on the file system.

```
ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/tmp/' identified by "<password>";
```

2. Merge the encryption keys from the source into the temporary encryption wallet.

```
ADMINISTER KEY MANAGEMENT MERGE KEYSTORE '+DATAC1/ODA/orawallet/tde/' INTO EXISTING KEYSTORE
'/tmp/' IDENTIFIED BY "<password>" WITH BACKUP;
```

3. If required, copy the temporary encryption wallet to a file system location that can be accessed from the target system.

```
[oracle@myhost1 ~]$ scp /tmp/ewallet.p12 myhost2:/tmp
```

4. Configure SQL*Net and the Oracle Listener on the source and target, and verify that the connections work in both directions.
    A. Create a `listener.ora` file on the source and start it.

```
[oracle@myhost1 ~]$ cat $ORACLE_HOME/network/admin/listener.ora
SID_LIST_LISTENERORG =
 (SID_LIST =
  (SID_DESC =
   (SID_NAME = ODA)
   (ORACLE_HOME = /oracle/ODA/19)
  )
 )
LISTENERORG =
 (DESCRIPTION_LIST =
  (DESCRIPTION =
```

```
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost1)(PORT = 1600))
  )
 )

[oracle@myhost1 ~]$ lsnrctl start LISTENERORG

LSNRCTL for Linux: Version 19.0.0.0.0 - Production on 11-JAN-2021 19:47:44

Copyright (c) 1991, 2020, Oracle.  All rights reserved.

Starting /oracle/ODA/19/bin/tnslsnr: please wait...

TNSLSNR for Linux: Version 19.0.0.0.0 - Production
System parameter file is /oracle/ODA/19/network/admin/listener.ora
Log messages written to /oracle/base/diag/tnslsnr/migtest1/listenerorg/alert/log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=myhost1)(PORT=1600)))

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=myhost1)(PORT=1600)))
STATUS of the LISTENER
------------------------
Alias                     LISTENERORG
Version                   TNSLSNR for Linux: Version 19.0.0.0.0 - Production
Start Date                11-JAN-2021 19:47:44
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /oracle/ODA/19/network/admin/listener.ora
Listener Log File         /oracle/base/diag/tnslsnr/migtest1/listenerorg/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=myhost1)(PORT=1600)))
Services Summary...
Service "ODA" has 1 instance(s).
  Instance "ODA", status UNKNOWN, has 1 handler(s) for this service...
The command completed successfully
```

B. Configure a `tnsnames.ora` file.

```
[oracle@myhost1 ~]$ cat $ORACLE_HOME/network/admin/tnsnames.ora
ORGODA =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost1)(PORT = 1600))
  )
  (CONNECT_DATA =
   (SERVICE_NAME = ODA)
  )
 )
DUPODA =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost2)(PORT = 1600))
  )
  (CONNECT_DATA =
   (SERVICE_NAME = ODA)
  )
 )
```

On the target system, perform the following steps:

1. Create an Oracle password file under `$ORACLE_HOME/dbs`.

```
[oracle@myhost2 dbs]$ orapwd file=orapwODA password=__Oracle123 force=y
```

2. Prepare an Oracle initialization parameter file (`pfile`) to start the target database instance into the `nomount` state.

```
[oracle@myhost2 dbs]$ cat initODA.ora
_kolfuseslf=TRUE
_disable_directory_link_check=TRUE
db_name='ODA'
memory_max_target=10G
memory_target=8G
tde_configuration='KEYSTORE_CONFIGURATION=FILE'
wallet_root='+DATAC1/ODA/orawallet'
control_files='+DATAC1/ODA/cntrlODA.dbf'
cluster_database='FALSE'
```

3. Configure SQL*Net and the Oracle Listener on the source and target and verify that the connections work in both directions.

   A. Create a `listener.ora` file on the target and start it.

```
[oracle@myhost2 dbs]$ cat $ORACLE_HOME/network/admin/listener.ora
SID_LIST_LISTENERDUP =
 (SID_LIST =
  (SID_DESC =
   (SID_NAME = ODA)
   (ORACLE_HOME = /oracle/ODA/19)
  )
 )
LISTENERDUP =
 (DESCRIPTION_LIST =
  (DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost2)(PORT = 1600))
  )
 )

[oracle@myhost2 dbs]$ lsnrctl start LISTENERDUP

LSNRCTL for Linux: Version 19.0.0.0.0 - Production on 11-JAN-2021 19:50:43

Copyright (c) 1991, 2020, Oracle.  All rights reserved.

Starting /oracle/ODA/19/bin/tnslsnr: please wait...

TNSLSNR for Linux: Version 19.0.0.0.0 - Production
System parameter file is /oracle/ODA/19/network/admin/listener.ora
Log messages written to /oracle/base/diag/tnslsnr/migtest2/listenerdup/alert/log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=myhost2)(PORT=1600)))

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=myhost2)(PORT=1600)))
STATUS of the LISTENER
------------------------
Alias                     LISTENERDUP
Version                   TNSLSNR for Linux: Version 19.0.0.0.0 - Production
Start Date                11-JAN-2021 19:50:43
```

```
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /oracle/ODA/19/network/admin/listener.ora
Listener Log File         /oracle/base/diag/tnslsnr/migtest2/listenerdup/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=myhost2)(PORT=1600)))
Services Summary...
Service "ODA" has 1 instance(s).
  Instance "ODA", status UNKNOWN, has 1 handler(s) for this service...
The command completed successfully
```

B. Configure a `tnsnames.ora` file.

```
[oracle@myhost2 ~]$ cat $ORACLE_HOME/network/admin/tnsnames.ora
ORGODA =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost1)(PORT = 1600))
  )
  (CONNECT_DATA =
   (SERVICE_NAME = ODA)
  )
 )
DUPODA =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost2)(PORT = 1600))
  )
  (CONNECT_DATA =
   (SERVICE_NAME = ODA)
  )
 )
```

4. Create the required directory for the encryption wallet in ASM.

```
[oracle@myhost2 ~]$ asmcmd
ASMCMD> cd +DATAC1
ASMCMD> mkdir ODA
ASMCMD> cd ODA
ASMCMD> mkdir orawallet
ASMCMD> mkdir orawallet/tde
```

5. Start the target instance into the `nomount` state by using the prepared `pfile`.

```
SQL> startup nomount pfile='?/dbs/initODA.ora"
ORACLE instance started.
```

6. Create an empty encryption wallet on ASM on the target system.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE IDENTIFIED BY "<password>";

keystore altered.
```

7. Merge the encryption keys from the previously created temporary encryption wallet into the new encryption wallet on ASM, and open it.

```
SQL> ADMINISTER KEY MANAGEMENT MERGE KEYSTORE '/tmp/' IDENTIFIED BY "<password>" INTO EXISTING
KEYSTORE '+DATAC1/ODA/orawallet/tde/' IDENTIFIED BY "<password>" WITH BACKUP;

keystore altered.

SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "<password>";

keystore altered.
```

8. Restart the target instance into the `nomount` state by using the prepared `pfile`.
9. Open the encryption wallet and create an autologin wallet.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "<password>";

keystore altered.
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE IDENTIFIED BY
"<password>";

keystore altered.
```

10. Restart the target instance into the `nomount` state by using the prepared `pfile` and verify that the encryption wallet opens automatically.

```
SQL> select WRL_TYPE,STATUS,WALLET_TYPE,WALLET_ORDER,KEYSTORE_MODE from v$encryption_wallet;

WRL_TYPE             STATUS                         WALLET_TYPE          WALLET_OR KEYSTORE
-------------------- ------------------------------ -------------------- --------- --------
FILE                 OPEN                           AUTOLOGIN            SINGLE    NONE
```

On the source system, run duplication of the source database.

```
connect target sys/"__Oracle123"@ORGODA
connect auxiliary sys/"__Oracle123"@DUPODA

run {
ALLOCATE CHANNEL t1 DEVICE TYPE disk;
ALLOCATE CHANNEL t2 DEVICE TYPE disk;
ALLOCATE CHANNEL t3 DEVICE TYPE disk;
ALLOCATE CHANNEL t4 DEVICE TYPE disk;
ALLOCATE CHANNEL t5 DEVICE TYPE disk;
ALLOCATE CHANNEL t6 DEVICE TYPE disk;
ALLOCATE CHANNEL t7 DEVICE TYPE disk;
ALLOCATE CHANNEL t8 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a1 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a2 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a3 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a4 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a5 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a6 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a7 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a8 DEVICE TYPE disk;
duplicate target database to ODA from active database using backupset nofilenamecheck;
}
```

Example 2: RMAN Duplicate from an Unencrypted Oracle 19c Source Database with Existing TDE Configuration

This example illustrates the RMAN Duplicate Database migration method in detail by migrating an unencrypted source database, ODA, from an on-premises Exadata to Exadata Cloud@Customer. The source database has an existing TDE configuration (encryption wallet).

On the source system, perform the following steps:

1. Create a temporary encryption wallet on the file system.

```
ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/tmp/' identified by "<password>";
```

2. Merge the encryption keys from the source into the temporary encryption wallet.

```
ADMINISTER KEY MANAGEMENT MERGE KEYSTORE '+DATAC1/ODA/orawallet/tde/' INTO EXISTING KEYSTORE
'/tmp/' IDENTIFIED BY "<password>" WITH BACKUP;
```

3. If required, copy the temporary encryption wallet to a file system location that can be accessed from the target system.

```
[oracle@myhost1 ~]$ scp /tmp/ewallet.p12 myhost2:/tmp
```

4. Configure SQL*Net and the Oracle Listener on the source and target, and verify that the connections work in both directions.

   A. Create a `listener.ora` file on the source and start it.

```
[oracle@myhost1 ~]$ cat $ORACLE_HOME/network/admin/listener.ora
SID_LIST_LISTENERORG =
 (SID_LIST =
  (SID_DESC =
   (SID_NAME = ODA)
   (ORACLE_HOME = /oracle/ODA/19)
  )
 )
LISTENERORG =
 (DESCRIPTION_LIST =
  (DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost1)(PORT = 1600))
  )
 )

[oracle@myhost1 ~]$ lsnrctl start LISTENERORG

LSNRCTL for Linux: Version 19.0.0.0.0 - Production on 11-JAN-2021 19:47:44

Copyright (c) 1991, 2020, Oracle.  All rights reserved.

Starting /oracle/ODA/19/bin/tnslsnr: please wait...

TNSLSNR for Linux: Version 19.0.0.0.0 - Production
System parameter file is /oracle/ODA/19/network/admin/listener.ora
Log messages written to /oracle/base/diag/tnslsnr/migtest1/listenerorg/alert/log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=myhost1)(PORT=1600)))

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=myhost1)(PORT=1600)))
STATUS of the LISTENER
------------------------
Alias                     LISTENERORG
Version                   TNSLSNR for Linux: Version 19.0.0.0.0 - Production
```

```
Start Date                11-JAN-2021 19:47:44
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /oracle/ODA/19/network/admin/listener.ora
Listener Log File         /oracle/base/diag/tnslsnr/migtest1/listenerorg/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=myhost1)(PORT=1600)))
Services Summary...
Service "ODA" has 1 instance(s).
  Instance "ODA", status UNKNOWN, has 1 handler(s) for this service...
The command completed successfully
```

B. Configure a `tnsnames.ora` file.

```
[oracle@myhost1 ~]$ cat $ORACLE_HOME/network/admin/tnsnames.ora
ORGODA =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost1)(PORT = 1600))
  )
  (CONNECT_DATA =
   (SERVICE_NAME = ODA)
  )
 )
DUPODA =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost2)(PORT = 1600))
  )
  (CONNECT_DATA =
   (SERVICE_NAME = ODA)
  )
 )
```

On the target system, perform the following steps:

1. Create an Oracle password file under `$ORACLE_HOME/dbs`.

```
[oracle@myhost2 dbs]$ orapwd file=orapwODA password=__Oracle123 force=y
```

2. Prepare an Oracle initialization parameter file (`pfile`) to start the target database instance into the `nomount` state.

```
[oracle@myhost2 dbs]$ cat initODA.ora
_kolfuseslf=TRUE
_disable_directory_link_check=TRUE
db_name='ODA'
memory_max_target=10G
memory_target=8G
tde_configuration='KEYSTORE_CONFIGURATION=FILE'
wallet_root='+DATAC1/ODA/orawallet'
control_files='+DATAC1/ODA/cntrlODA.dbf'
cluster_database='FALSE'
```

3. Configure SQL*Net and the Oracle Listener on the source and target, and verify that the connections work in both directions.

   A. Create a `listener.ora` file on the target and start it.

```
[oracle@myhost2 dbs]$ cat $ORACLE_HOME/network/admin/listener.ora
SID_LIST_LISTENERDUP =
 (SID_LIST =
  (SID_DESC =
   (SID_NAME = ODA)
   (ORACLE_HOME = /oracle/ODA/19)
  )
 )
LISTENERDUP =
 (DESCRIPTION_LIST =
  (DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost2)(PORT = 1600))
  )
 )

[oracle@myhost2 dbs]$ lsnrctl start LISTENERDUP

LSNRCTL for Linux: Version 19.0.0.0.0 - Production on 11-JAN-2021 19:50:43

Copyright (c) 1991, 2020, Oracle.  All rights reserved.

Starting /oracle/ODA/19/bin/tnslsnr: please wait...

TNSLSNR for Linux: Version 19.0.0.0.0 - Production
System parameter file is /oracle/ODA/19/network/admin/listener.ora
Log messages written to /oracle/base/diag/tnslsnr/migtest2/listenerdup/alert/log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=myhost2)(PORT=1600)))

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=myhost2)(PORT=1600)))
STATUS of the LISTENER
------------------------
Alias                     LISTENERDUP
Version                   TNSLSNR for Linux: Version 19.0.0.0.0 - Production
Start Date                11-JAN-2021 19:50:43
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /oracle/ODA/19/network/admin/listener.ora
Listener Log File         /oracle/base/diag/tnslsnr/migtest2/listenerdup/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=myhost2)(PORT=1600)))
Services Summary...
Service "ODA" has 1 instance(s).
  Instance "ODA", status UNKNOWN, has 1 handler(s) for this service...
The command completed successfully
```

B. Configure a `tnsnames.ora` file.

```
[oracle@myhost2 ~]$ cat $ORACLE_HOME/network/admin/tnsnames.ora
ORGODA =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost1)(PORT = 1600))
  )
  (CONNECT_DATA =
   (SERVICE_NAME = ODA)
  )
 )
DUPODA =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost2)(PORT = 1600))
  )
  (CONNECT_DATA =
   (SERVICE_NAME = ODA)
  )
 )
```

4. Create the required directory for the encryption wallet in ASM.

```
[oracle@myhost2 ~]$ asmcmd
ASMCMD> cd +DATAC1
ASMCMD> mkdir ODA
ASMCMD> cd ODA
ASMCMD> mkdir orawallet
ASMCMD> mkdir orawallet/tde
```

5. Start the target instance into the `nomount` state by using the prepared `pfile`.

```
SQL> startup nomount pfile='?/dbs/initODA.ora"
ORACLE instance started.
```

6. Create an empty encryption wallet on ASM on the target system.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE IDENTIFIED BY "<password>";

keystore altered.
```

7. Merge the encryption keys from the previously created temporary encryption wallet into the new encryption wallet on ASM, and open it.

```
SQL> ADMINISTER KEY MANAGEMENT MERGE KEYSTORE '/tmp/' IDENTIFIED BY "<password>" INTO EXISTING
KEYSTORE '+DATAC1/ODA/orawallet/tde/' IDENTIFIED BY "<password>" WITH BACKUP;

keystore altered.

SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "<password>";

keystore altered.
```

8. Restart the target instance into the `nomount` state by using the prepared `pfile`.
9. Open the encryption wallet and create an autologin wallet.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "<password>";

keystore altered.
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE IDENTIFIED BY
"<password>";

keystore altered.
```

10. Restart the target instance into the `nomount` state by using the prepared `pfile` and verify that the encryption wallet opens automatically.

```
SQL> select WRL_TYPE,STATUS,WALLET_TYPE,WALLET_ORDER,KEYSTORE_MODE from v$encryption_wallet;

WRL_TYPE             STATUS                         WALLET_TYPE          WALLET_OR KEYSTORE
-------------------- ------------------------------ -------------------- --------- --------
FILE                 OPEN                           AUTOLOGIN            SINGLE    NONE
```

On the source system, run duplication of the source database.

```
connect target sys/"__Oracle123"@ORGODA
connect auxiliary sys/"__Oracle123"@DUPODA

run {
ALLOCATE CHANNEL t1 DEVICE TYPE disk;
ALLOCATE CHANNEL t2 DEVICE TYPE disk;
ALLOCATE CHANNEL t3 DEVICE TYPE disk;
ALLOCATE CHANNEL t4 DEVICE TYPE disk;
ALLOCATE CHANNEL t5 DEVICE TYPE disk;
ALLOCATE CHANNEL t6 DEVICE TYPE disk;
ALLOCATE CHANNEL t7 DEVICE TYPE disk;
ALLOCATE CHANNEL t8 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a1 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a2 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a3 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a4 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a5 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a6 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a7 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a8 DEVICE TYPE disk;
duplicate target database to ODA from active database using backupset as encrypted nofilenamecheck;
}
```

Example 3: RMAN Duplicate from an Unencrypted Oracle 19c Source Database with No TDE Configuration

This example illustrates the RMAN Duplicate Database migration method in detail by migrating an unencrypted source database, ODA, from an on-premises Exadata to Exadata Cloud@Customer. The source database has no existing TDE configuration (encryption wallet). TDE encryption is implemented on the target after duplication.

On the source system, configure SQL*Net and the Oracle Listener on the source and target and verify that the connections work in both directions:

1. Create a `listener.ora` file on the source and start it.

```
[oracle@myhost1 ~]$ cat $ORACLE_HOME/network/admin/listener.ora
SID_LIST_LISTENERORG =
```

```
 (SID_LIST =
  (SID_DESC =
   (SID_NAME = ODA)
   (ORACLE_HOME = /oracle/ODA/19)
  )
 )
LISTENERORG =
 (DESCRIPTION_LIST =
  (DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost1)(PORT = 1600))
  )
 )

[oracle@myhost1 ~]$ lsnrctl start LISTENERORG

LSNRCTL for Linux: Version 19.0.0.0.0 - Production on 11-JAN-2021 19:47:44

Copyright (c) 1991, 2020, Oracle.  All rights reserved.

Starting /oracle/ODA/19/bin/tnslsnr: please wait...

TNSLSNR for Linux: Version 19.0.0.0.0 - Production
System parameter file is /oracle/ODA/19/network/admin/listener.ora
Log messages written to /oracle/base/diag/tnslsnr/migtest1/listenerorg/alert/log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=myhost1)(PORT=1600)))

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=myhost1)(PORT=1600)))
STATUS of the LISTENER
------------------------
Alias                     LISTENERORG
Version                   TNSLSNR for Linux: Version 19.0.0.0.0 - Production
Start Date                11-JAN-2021 19:47:44
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /oracle/ODA/19/network/admin/listener.ora
Listener Log File         /oracle/base/diag/tnslsnr/migtest1/listenerorg/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=myhost1)(PORT=1600)))
Services Summary...
Service "ODA" has 1 instance(s).
  Instance "ODA", status UNKNOWN, has 1 handler(s) for this service...
The command completed successfully
```

2. Configure a `tnsnames.ora` file.

```
[oracle@myhost1 ~]$ cat $ORACLE_HOME/network/admin/tnsnames.ora
ORGODA =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost1)(PORT = 1600))
  )
  (CONNECT_DATA =
   (SERVICE_NAME = ODA)
  )
 )
```

```
DUPODA =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost2)(PORT = 1600))
  )
  (CONNECT_DATA =
   (SERVICE_NAME = ODA)
  )
 )
```

On the target system, perform the following steps:

1. Create an Oracle password file under `$ORACLE_HOME/dbs`.

```
[oracle@myhost2 dbs]$ orapwd file=orapwODA password=__Oracle123 force=y
```

2. Prepare an Oracle initialization parameter file (`pfile`) to start the target database instance into the `nomount` state.

```
[oracle@myhost2 dbs]$ cat initODA.ora
_kolfuseslf=TRUE
_disable_directory_link_check=TRUE
db_name='ODA'
memory_max_target=10G
memory_target=8G
tde_configuration='KEYSTORE_CONFIGURATION=FILE'
wallet_root='+DATAC1/ODA/orawallet'
control_files='+DATAC1/ODA/cntrlODA.dbf'
cluster_database='FALSE'
```

3. Configure SQL*Net and the Oracle Listener on the source and target and verify that the connections work in both directions.

   A. Create a `listener.ora` file on the target and start it.

```
[oracle@myhost2 dbs]$ cat $ORACLE_HOME/network/admin/listener.ora
SID_LIST_LISTENERDUP =
 (SID_LIST =
  (SID_DESC =
   (SID_NAME = ODA)
   (ORACLE_HOME = /oracle/ODA/19)
  )
 )
LISTENERDUP =
 (DESCRIPTION_LIST =
  (DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost2)(PORT = 1600))
  )
 )

[oracle@myhost2 dbs]$ lsnrctl start LISTENERDUP

LSNRCTL for Linux: Version 19.0.0.0.0 - Production on 11-JAN-2021 19:50:43

Copyright (c) 1991, 2020, Oracle.  All rights reserved.

Starting /oracle/ODA/19/bin/tnslsnr: please wait...

TNSLSNR for Linux: Version 19.0.0.0.0 - Production
System parameter file is /oracle/ODA/19/network/admin/listener.ora
```

```
Log messages written to /oracle/base/diag/tnslsnr/migtest2/listenerdup/alert/log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=myhost2)(PORT=1600)))

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=myhost2)(PORT=1600)))
STATUS of the LISTENER
------------------------
Alias                     LISTENERDUP
Version                   TNSLSNR for Linux: Version 19.0.0.0.0 - Production
Start Date                11-JAN-2021 19:50:43
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /oracle/ODA/19/network/admin/listener.ora
Listener Log File         /oracle/base/diag/tnslsnr/migtest2/listenerdup/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=myhost2)(PORT=1600)))
Services Summary...
Service "ODA" has 1 instance(s).
  Instance "ODA", status UNKNOWN, has 1 handler(s) for this service...
The command completed successfully
```

B. Configure a `tnsnames.ora` file.

```
[oracle@myhost2 ~]$ cat $ORACLE_HOME/network/admin/tnsnames.ora
ORGODA =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost1)(PORT = 1600))
  )
  (CONNECT_DATA =
   (SERVICE_NAME = ODA)
  )
 )
DUPODA =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost2)(PORT = 1600))
  )
  (CONNECT_DATA =
   (SERVICE_NAME = ODA)
  )
 )
```

4. Start the target instance into the `nomount` state by using the prepared `pfile`.

```
SQL> startup nomount pfile='?/dbs/initODA.ora"
ORACLE instance started.
```

On the source system, run duplication of the source database.

```
connect target sys/"__Oracle123"@ORGODA
connect auxiliary sys/"__Oracle123"@DUPODA

run {
ALLOCATE CHANNEL t1 DEVICE TYPE disk;
ALLOCATE CHANNEL t2 DEVICE TYPE disk;
ALLOCATE CHANNEL t3 DEVICE TYPE disk;
ALLOCATE CHANNEL t4 DEVICE TYPE disk;
```

```
ALLOCATE CHANNEL t5 DEVICE TYPE disk;
ALLOCATE CHANNEL t6 DEVICE TYPE disk;
ALLOCATE CHANNEL t7 DEVICE TYPE disk;
ALLOCATE CHANNEL t8 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a1 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a2 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a3 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a4 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a5 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a6 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a7 DEVICE TYPE disk;
ALLOCATE AUXILIARY CHANNEL a8 DEVICE TYPE disk;
duplicate target database to ODA from active database using backupset nofilenamecheck;
}
```

On the target system, perform the following steps:

1. Create the required directory for the encryption wallet in ASM.

```
[oracle@myhost2 ~]$ asmcmd
ASMCMD> cd +DATAC1
ASMCMD> mkdir ODA
ASMCMD> cd ODA
ASMCMD> mkdir orawallet
ASMCMD> mkdir orawallet/tde
```

2. Restart the target database into the open state.
3. Create an empty encryption wallet on ASM on the target system.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE IDENTIFIED BY "<password>";

keystore altered.
```

4. Open the encryption wallet and set a new master key.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "<password>";

keystore altered.
SQL> alter system set "_db_discard_lost_masterkey"=TRUE;

System altered.
SQL> ADMINISTER KEY MANAGEMENT SET KEY USING TAG 'MASTERKEY' FORCE KEYSTORE IDENTIFIED BY
"<password>" WITH BACKUP;

keystore altered.
```

5. Restart the target database into the open state.
6. Open the encryption wallet and create an autologin wallet.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "<password>";

keystore altered.
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE IDENTIFIED BY
"<password>";

keystore altered.
```

7. Restart the target instance into the open state by using the prepared `pfile` and verify that the encryption wallet opens automatically.

```
SQL> select WRL_TYPE,STATUS,WALLET_TYPE,WALLET_ORDER,KEYSTORE_MODE from v$encryption_wallet;

WRL_TYPE             STATUS                         WALLET_TYPE          WALLET_OR KEYSTORE
-------------------- ------------------------------ -------------------- --------- --------
FILE                 OPEN                           AUTOLOGIN            SINGLE    NONE
```

8. Shut down the database again.
9. For offline encryption of the whole database, follow these steps. If you want to perform online encryption, skip to step 10.
   A. Encrypt the SYSTEM and UNDO tablespaces in the `mount` state.
   B. Open the database.
   C. Set all the remaining permanent tablespaces offline.
   D. Start an encryption job for each data file.

```
shutdown immediate;
startup mount;

ALTER TABLESPACE SYSTEM ENCRYPTION OFFLINE ENCRYPT;
ALTER TABLESPACE PSAPUNDO001 ENCRYPTION OFFLINE ENCRYPT;
ALTER TABLESPACE PSAPUNDO002 ENCRYPTION OFFLINE ENCRYPT;
ALTER DATABASE OPEN;

declare
  cmd1 varchar2(32767);
  cmd2 varchar2(32767);
  job_cmd varchar2(32767);
  jobname varchar2(32767);
  prefix varchar2(512);
begin
  for t1 in (SELECT tablespace_name from dba_tablespaces where contents<>'TEMPORARY' and
contents<>'UNDO' and tablespace_name<>'SYSTEM') loop
    cmd1:='ALTER TABLESPACE '|| t1.tablespace_name ||' OFFLINE';
    begin
      execute immediate cmd1;
      exception
        when others then null;
    end;

    for f1 in (select file_name from dba_data_files where tablespace_name =
t1.tablespace_name) loop
      cmd2:='ALTER DATABASE DATAFILE '||''''||''''||f1.file_name||''''||''''||' ENCRYPT';

      job_cmd:='BEGIN EXECUTE IMMEDIATE '''||cmd2||'''; END;';

      prefix:=substr(t1.tablespace_name||'ENC',1,18);
      jobname:=dbms_scheduler.generate_job_name(prefix);
      dbms_scheduler.create_job(job_name => jobname,job_type => 'PLSQL_BLOCK',job_action =>
job_cmd,enabled => FALSE);
      dbms_scheduler.run_job(job_name => jobname,use_current_session => FALSE);
---      dbms_output.put_line(job_cmd);
    end loop;
```

```
  end loop;
end;
/
You can monitor the process by using the OS utility top, which shows numerous job queue
processes consuming CPU time for encryption; by using V$ENCRYPTED_TABLESPACES; or by
checking the status of the generated jobs in DBA_SCHEDULER_JOBS. For example:
Output of top:
376750 oracle    20   0   26.4g  99580  89440 S  26.8  0.1   0:02.93 ora_j007_MFG001
376512 oracle    20   0   26.7g  99684  89536 S  25.8  0.1   0:03.01 ora_j004_MFG001
376488 oracle    20   0   26.7g 106832  95292 S  24.8  0.1   0:03.12 ora_j000_MFG001
378349 oracle    20   0   26.7g  99120  89060 R  24.8  0.1   0:02.78 ora_j008_MFG001
378357 oracle    20   0   26.7g  99588  89432 R  24.5  0.1   0:02.46 ora_j00b_MFG001
376499 oracle    20   0   26.4g  98892  88844 R  22.5  0.1   0:02.85 ora_j003_MFG001
376495 oracle    20   0   26.7g  99832  89664 S  21.9  0.1   0:02.93 ora_j002_MFG001
376490 oracle    20   0   26.7g  99612  89472 S  21.2  0.1   0:02.58 ora_j001_MFG001
378360 oracle    20   0   26.4g 102676  92488 S  20.3  0.1   0:02.53 ora_j00c_MFG001
378353 oracle    20   0   26.7g  99784  89632 S  19.9  0.1   0:02.27 ora_j009_MFG001
376642 oracle    20   0   26.7g  99244  89164 S  19.6  0.1   0:02.82 ora_j006_MFG001
376558 oracle    20   0   26.4g  99284  89132 S  18.0  0.1   0:02.46 ora_j005_MFG001

Out of of V$ENCRYPTED_TABLESPACES:

SQL> select TS#,STATUS,ENCRYPTIONALG from V$ENCRYPTED_TABLESPACES;

       TS# STATUS     ENCRYPT
---------- ---------- -------
         0 NORMAL     AES128
         1 NORMAL     AES128
         2 NORMAL     AES128
         4 ENCRYPTING AES128
         5 ENCRYPTING AES128
         6 NORMAL     AES128
         7 NORMAL     AES128
         8 NORMAL     AES128
```

E. When all the data files are encrypted, set all the tablespaces back online. For example:

```
declare
  cmd1 varchar2(32767);
begin
  for t1 in (SELECT tablespace_name from dba_tablespaces where contents='PERMANENT' and
tablespace_name<>'SYSTEM') loop
    cmd1:='ALTER TABLESPACE '|| t1.tablespace_name ||' ONLINE';
    begin
      execute immediate cmd1;
      exception
        when others then null;
    end;
  end loop;
end;
/
```

10. For online encryption of the whole database, run the following PL/SQL script. Modify it to match your needs before running it.

```
declare
  old_file varchar2(500);
  new_file varchar2(500);
  cmd varchar2(32767);
  job_cmd varchar2(32767);
  jobname varchar2(32767);
  file_nr number;
  suffix varchar2(16):='_enc';
  prefix varchar2(512);
begin
  for t1 in (select tablespace_name from dba_tablespaces where contents<>'TEMPORARY') loop
    file_nr := 1;
    cmd := 'ALTER TABLESPACE '||t1.tablespace_name||' ENCRYPTION ONLINE ENCRYPT
FILE_NAME_CONVERT=(';
    for f1 in (select file_name from dba_data_files where tablespace_name = t1.tablespace_name)
loop
      if file_nr>1 then
        cmd:=cmd||',';
      end if;
      old_file := f1.file_name;
      if instr(old_file,'.')>1 then
        new_file := substr(old_file,1,instr(old_file,'.')-
1)||lpad(to_char(file_nr),3,'0')||suffix||'.dbf';
      else
        new_file := old_file||lpad(to_char(file_nr),3,'0')||suffix||'.dbf';
      end if;
      file_nr:=file_nr+1;
      cmd:=cmd||''''||''''||old_file||''''||''''||',''||''''||''''||new_file||''''||'''';
    end loop;
    cmd:=cmd||')';

    job_cmd:='BEGIN EXECUTE IMMEDIATE '''||cmd||'''; END;';

    prefix:=substr(t1.tablespace_name||'ENC',1,18);
    jobname:=dbms_scheduler.generate_job_name(prefix);
    dbms_scheduler.create_job(job_name => jobname,job_type => 'PLSQL_BLOCK',job_action =>
job_cmd,enabled => FALSE);
    dbms_scheduler.run_job(job_name => jobname,use_current_session => FALSE);
  end loop;
end;
/
```

11. Create an encrypted temporary tablespace to replace your unencrypted PSAPTEMP tablespace.

```
CREATE TEMPORARY TABLESPACE PSAPTEMP_ENC TEMPFILE '+DATAC2' SIZE 32767M ENCRYPTION ENCRYPT;
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE PSAPTEMP_ENC;
DROP TABLESPACE PSAPTEMP;
ALTER TABLESPACE PSAPTEMP_ENC RENAME TO PSAPTEMP;
```

Post-Migration Tasks

After you migrate the database to the target system, there are usually numerous post-migration tasks that you need to perform on the Oracle Database side and on the SAP application side.

Database-Related Post-Migration Tasks

This section describes some common Oracle Database–related post-migration tasks.

Adjust Oracle Initialization Parameters

Follow the relevant SAP Notes for your target system and set Oracle initialization parameters appropriately. Keep the server parameter file (`spfile`) on ASM to be shared across the database instances.

Adjust Oracle RAC–specific parameters. For example, if your target system has two database compute nodes, make the following adjustments:

```
ODA001.instance_number=1
ODA002.instance_number=2
ODA001.thread=1
ODA002.thread=2
ODA001.undo_tablespace='PSAPUNDO1'
ODA002.undo_tablespace='PSAPUNDO2'
*.remote_listener='<your_scan_name>:1521'
```

If you are using Oracle Database 19c, ensure that parameters for TDE are set properly for each database instance.

```
*.tde_configuration = 'KEYSTORE_CONFIGURATION=FILE'
*.wallet_root = '+DATAC1/ODA/orawallet'
```

For Oracle Database 12.1, 12.2, or 18c, TDE is configured in `sqlnet.ora` instead of the `spfile`. For example:

```
ENCRYPTION_WALLET_LOCATION=
 (SOURCE=
  (METHOD=FILE)
   (METHOD_DATA=
    (DIRECTORY=+DATAC1/ODA/orawallet)))
```

The following additional initialization parameters might also need adjustment:

```
log_archive_dest
db_create_file_dest
db_create_online_log_dest_1
db_create_online_log_dest_2
compatible
db_recovery_file_dest
db_recovery_file_dest_size
```

Adjust Online Redo Logs and UNDO Tablespaces

Start your first instance and add online redo log groups and UNDO tablespaces as needed.

Manage the Password File

Check whether your version of Oracle Database supports shared password files on ASM. If so, we recommend using a shared password file across the database instances. Otherwise, or if you do not want the password file to be shared, place it under `$ORACLE_HOME/dbs`. In this case, ensure that the proper password file is copied to all other Oracle Database homes on all database compute nodes.

Configure the Database with Oracle Clusterware

As mentioned earlier, this document assumes that an initial installation with SWPM is performed and that the initial database is then replaced by the migrated one. This process configures the new temporary database with Oracle Clusterware. SAP application servers installed using SWPM also have their own dedicated database service.

If you could not follow this approach, you can perform all the required steps manually by using the `srvctl` utility.

1. Register the database and the database instances with Oracle Clusterware.
2. Create a database service for each application server that connects to the system.
3. Configure the database services to run only on one database instance at a time to ensure that each SAP application server connects to only one database instance at the same time.
4. Configure the database services to fail over to other database instances as needed (but always to only one).

SAP-Related Post-Migration Tasks

This section describes some common SAP NetWeaver–related post-migration tasks.

Application Server Profiles

Application server profiles are created during application server installation using SWPM. If you install new application servers after migration, SWPM automatically creates the proper profiles. SWPM also creates a Linux command line script that you must run on the target system to create a database service dedicated to the application server being installed.

If you keep your current application servers, you need to modify the application server instance profiles so that each application server connects to its dedicated database server through the SCAN listener. Also, you must create the database services.

For example, if you want to use EZCONNECT for a dialog instance named D01 and a database service named ODA_D01:

```
SAP instance profile parameters:

SAPSYSTEMNAME=ODA
INSTANCE_NAME=D01
dbs/ora/tnsname=//<scan-listener>/ODA_D01
```

If you want to use `sqlnet.ora` with a TNS name to connect:

```
Sqlnet.ora:
ODA_D01= (DESCRIPTION =
   (ADDRESS=(PROTOCOL=TCP)
            (HOST=<scan-listener>)
            (PORT=1521)
   )
   (CONNECT_DATA =
       (SERVICE_NAME = ODA_D01)
       (GLOBAL_NAME = ODA)
       (FAILOVER_MODE =
            (TYPE = SELECT)
            (METHOD = BASIC)
      )
   )
)
```

```
SAP instance profile parameters:

SAPSYSTEMNAME=ODA
INSTANCE_NAME=D01
dbs/ora/tnsname=ODA_D01
```

Other SAP Post-Migration Tasks

Following are some other potential SAP-related post-migration tasks:

- Adjust SAP RFC connections.
- Reschedule SAP jobs.
- Make any necessary SAP Solution Manager changes.
- Install new SAP licenses.
- Configure remote backups using DB13.
- For Oracle RAC, adjust SAP application server profiles as recommended.

References

SAP

Most of the SAP links require SAP login credentials for access.

SAP Documentation

- SAP Product Availability Matrix (PAM)
- SAP Software Logistics Toolset (SL Tools)
- SAP Download Manager
- SAP Software Download Center (SWDC)
- SAP NetWeaver Guide Finder
- SAP Community Network: Oracle Community
- SAP Help Portal: TCP/IP Ports of All SAP Products

SAP Notes

- 2956661 - SAP NetWeaver on Oracle Database Exadata Cloud@Customer
- 2614080 - SAP on Linux with Oracle Database Exadata Cloud@Customer: Enhanced Monitoring
- 2470718 - Oracle Database Parameter 12.2 / 18c / 19c
- 1888485 - Database Parameter for 12.1.0.2
- 2378252 - Oracle Database Initialization Parameters for SAP NetWeaver Systems
- 2520061 - SAP on Oracle Cloud Infrastructure: Support prerequisites
- 611361 - Hostnames of SAP ABAP Platform servers
- 146505 - SAP GUI for the Java Environment
- 1914631 - Central Technical Note for Oracle Database 12c Release 1 (12.1)

- 2470660 - Oracle Database Central Technical Note for 12c Release 2 (12.2)
- 2660020 - Central Technical Note for Oracle Database 18c
- 2799900 - Central Technical Note for Oracle Database 19c
- 1868094 - Overview: Oracle Security SAP Notes
- 1496927 - Protection of SAP instances through Oracle Clusterware
- 2591575 - Using Oracle Transparent Data Encryption (TDE) with SAP NetWeaver
- 2799991 - TDE Encryption Conversions for Tablespaces and Databases
- 1598594 - BR*Tools configuration for Oracle installation using user "oracle"
- 113747 - Owners and authorizations of BR*Tools
- 776505 - ORA-01017/ORA-01031 in BR*Tools on Linux and Solaris 11
- 2618837 - Oracle Exadata Cloud: Patches for 12.2.0.1
- 2618881 - Oracle Exadata Cloud: Patches for 12.1.0.2
- 2884306 - Managing SAPDATA_HOME and ORACLE_BASE on Oracle Engineered Systems
- 2992680 - Managing shared and multiple Oracle Homes on Oracle Engineered Systems
- 2660062 - Oracle Exadata Cloud Service: Patches for Oracle 18c
- 2799970 - Patches for 19c: Oracle Exadata Cloud
- 2422996 - Oracle: OPatch Versions 12.2.0.1.8, 11.2.0.3.18 and Newer
- 3018983 - Additional information to the Oracle technical brief "Migrating SAP NetWeaver based Systems to Oracle Exadata Cloud Solutions"
- 527843 - Oracle RAC support in the SAP environment
- 2072655 – FAQ: Oracle Timezone/DST in SAP systems
- 3110260 - Oracle: SAP Bundle Patches and Oracle Database Migration

Oracle

- Oracle Database Exadata Cloud@Customer Gen 2
- Oracle Cloud Hosting and Delivery Policies
- Oracle Database
- Oracle Linux
- Oracle-SAP Solutions site

CONNECT WITH US

Call +1.800.ORACLE1 or visit oracle.com.
Outside North America, find your local office at oracle.com/contact.

blogs.oracle.com facebook.com/oracle twitter.com/oracle

Migrating SAP NetWeaver® Based Systems Within the Scope of Oracle Databases
February 2021
Author: Markus Breunig
Contributing authors: Torsten Grambs, Jan Klokkers, Andreas Becker, Kurt Broeg, Jens Schmidt