Oracle® Database

Migrating and Converting Non-CDBs to a PDB with a Different Endian Operating System

19c
F10904-01
January 2019

Oracle Database Migrating and Converting Non-CDBs to a PDB with a Different Endian Operating System, 19c

F10904-01

Primary Authors: Sunil Surabhi, Nirmal Kumar

Contributing Authors: Lance Ashdown, Padmaja Potineni, Rajesh Bhatiya, Prakash Jashnani, Douglas Williams, Mark Bauer

Contributors: Roy Swonger, Byron Motta, Hector Vieyra Farfan, Carol Tagliaferri, Mike Dietrich, Marcus Doeringer, Umesh Aswathnarayana Rao, Rae Burns, Subrahmanyam Kodavaluru, Cindy Lim, Amar Mbaye, Akash Pathak, Thomas Zhang, Zhihai Zhang

Contents

Preface

Use Case Scenario for this Document v
Documentation Accessibility vi

1 Transporting Databases

Transporting Data Across Platforms 1-1
General Limitations on Transporting Data 1-3
Compatibility Considerations for Transporting Data 1-5
Limitations on Full Transportable Export/import 1-6

2 Converting Data to the Endian Format of the Target Operating System

Converting Data Between Platforms 2-1
Converting Data Between Platforms Using the DBMS_FILE_TRANSFER Package 2-2
Converting Data Between Platforms Using RMAN 2-3
- Converting Tablespaces on the Source System After Export 2-4
- Converting Data Files on the Target System Before Import 2-5

3 Connecting to the Database with SQL*Plus

Step 1: Open a Command Window 3-1
Step 2: Set Operating System Environment Variables 3-1
Step 3: Start SQL*Plus 3-2
Step 4: Submit the SQL*Plus CONNECT Command 3-2

4 Migrating Oracle Database

Checking Character Set Compatibility 4-1
Transporting a Database Using an Export Dump File 4-1

Index

Preface

This guide provides a compilation of topics from the Oracle Database user assistance documentation that are collected to help you complete a specific use case scenario.

- Use Case Scenario for this Document
- Documentation Accessibility

Use Case Scenario for this Document

Use this scenario document to assist you to convert an earlier release non-CDB to a PDB, running on a different endian operating system. After that conversion, you can then plug that PDB into Oracle Database 19c.

This scenario applies to the following non-CDBs:

- Oracle Database 11g Release 2 (11.2.0.3)
- Oracle Database 11g Release 2 (11.2.0.4)
- Oracle Database 12c Release 1 (12.1.0.1)
- Oracle Database 12c Release 1 (12.1.0.2)
- Oracle Database 12c Release 2 (12.2.0.1)

Prerequisites for this Scenario

- The source non-CDB and the target CDB host must use compatible character sets and national character sets.
- You have created and configured Oracle Database 19c on the target CDB host, using the multitenant architecture as your configuration option, so that you have a CDB and a default PDB.
- You are prepared to place your user-defined tablespaces on the source non-CDB into read-only mode.

Outline for this Scenario

1. **Transporting Databases**. Transport your database to the new platform
2. **Converting Data to the Endian Format of the Target Operating System**. Convert your endian format, and then convert data to the new platform.
3. **Connecting to Oracle Database**
4. **Migrating Oracle Database**. Migrate the database to the PDB on the target multitenant architecture host.

These steps correspond to the chapters in this document.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

1 Transporting Databases

You can transport a database to a new Oracle Database instance.

- Transporting Data Across Platforms
  You can transport data across platforms.
- General Limitations on Transporting Data
  There are general limitations on transporting data. There are also limitations that are specific to full transportable export/import, transportable tablespaces, or transportable tables.
- Compatibility Considerations for Transporting Data
  When transporting data, Oracle Database computes the lowest compatibility level at which the target database must run.
- Limitations on Full Transportable Export/import
  There are limitations on full transportable export/import.

Transporting Data Across Platforms

You can transport data across platforms.

The functionality of transporting data across platforms can be used to:

- Enable a database to be migrated from one platform to another.
- Provide an easier and more efficient means for content providers to publish structured data and distribute it to customers running Oracle Database on different platforms.
- Simplify the distribution of data from a data warehouse environment to data marts, which are often running on smaller platforms.
- Enable the sharing of read-only tablespaces between Oracle Database installations on different operating systems or platforms, assuming that your storage system is accessible from those platforms and the platforms all have the same endianness, as described in the sections that follow.

Many, but not all, platforms are supported for cross-platform data transport. You can query the `V$TRANSPORTABLE_PLATFORM` view to see the platforms that are supported, and to determine each platform's endian format (byte ordering). The following query displays the platforms that support cross-platform data transport:

```
COLUMN PLATFORM_NAME FORMAT A40
COLUMN ENDIAN_FORMAT A14

SELECT PLATFORM_ID, PLATFORM_NAME, ENDIAN_FORMAT
  FROM V$TRANSPORTABLE_PLATFORM
  ORDER BY PLATFORM_ID;

PLATFORM_ID PLATFORM_NAME                              ENDIAN_FORMAT
----------- ---------------------------------------- --------------
          1 Solaris[tm] OE (32-bit)                  Big
          2 Solaris[tm] OE (64-bit)                  Big
```

```
 3 HP-UX (64-bit)                          Big
 4 HP-UX IA (64-bit)                       Big
 5 HP Tru64 UNIX                           Little
 6 AIX-Based Systems (64-bit)              Big
 7 Microsoft Windows IA (32-bit)           Little
 8 Microsoft Windows IA (64-bit)           Little
 9 IBM zSeries Based Linux                 Big
10 Linux IA (32-bit)                       Little
11 Linux IA (64-bit)                       Little
12 Microsoft Windows x86 64-bit            Little
13 Linux x86 64-bit                        Little
15 HP Open VMS                             Little
16 Apple Mac OS                            Big
17 Solaris Operating System (x86)          Little
18 IBM Power Based Linux                   Big
19 HP IA Open VMS                          Little
20 Solaris Operating System (x86-64)       Little
21 Apple Mac OS (x86-64)                   Little
```

If the source platform and the target platform are of the same endianness, then the data is transported from the source platform to the target platform without any data conversion.

If the source platform and the target platform are of different endianness, then the data being transported must be converted to the target platform format. You can convert the data using one of the following methods:

- The `GET_FILE` or `PUT_FILE` procedure in the `DBMS_FILE_TRANSFER` package

  When you use one of these procedures to move data files between the source platform and the target platform, each block in each data file is converted to the target platform's endianness. The conversion occurs on the target platform.

- The RMAN `CONVERT` command

  Run the RMAN `CONVERT` command on the source or target platform. This command converts the data being transported to the target platform format.

> **Note:**
>
> Conversion of data files between different endian formats is not supported for data files having undo segments.

Before the data in a data file can be transported to a different platform, the data file header must identify the platform to which it belongs. To transport read-only tablespaces between Oracle Database installations on different platforms, make the data file read/write at least once.

> **See Also:**
>
> "Converting Data Between Platforms"

General Limitations on Transporting Data

There are general limitations on transporting data. There are also limitations that are specific to full transportable export/import, transportable tablespaces, or transportable tables.

Be aware of the following general limitations as you plan to transport data:

- The source and the target databases must use compatible database character sets. Specifically, one of the following must be true:
  - The database character sets of the source and the target databases are the same.
  - The source database character set is a strict (binary) subset of the target database character set, and the following three conditions are true:
    - The source database is Oracle Database 10*g* Release 1 (10.1.0.3) or later.
    - The tablespaces to be transported contain no table columns with character length semantics or the maximum character width is the same in both the source and target database character sets.
    - The data to be transported contains no columns with the `CLOB` data type, or the source and the target database character sets are both single-byte or both multibyte.
  - The source database character set is a strict (binary) subset of the target database character set, and the following two conditions are true:
    - The source database is before Oracle Database 10*g* Release 1 (10.1.0.3).
    - The maximum character width is the same in the source and target database character sets.

> **Note:**
>
> The subset-superset relationship between character sets recognized by Oracle Database is documented in the *Oracle Database Globalization Support Guide*.

- The source and the target databases must use compatible national character sets. Specifically, one of the following must be true:
  - The national character sets of the source and target databases are the same.
  - The source database is Oracle Database 10*g* Release 1 (10.1.0.3) or later, and the tablespaces to be transported contain no columns with `NCHAR`, `NVARCHAR2`, or `NCLOB` data types.
- When running a transportable export operation, the following limitations apply:
  - The default tablespace of the user performing the export must not be one of the tablespaces being transported.
  - The default tablespace of the user performing the export must be writable.

- In a non-CDB, you cannot transport a tablespace to a target database that contains a tablespace of the same name.

  In a CDB, you cannot transport a tablespace to a target container that contains a tablespace of the same name. However, different containers can have tablespaces with the same name.

  You can use the `REMAP_TABLESPACE` import parameter to import the database objects into a different tablespace. Alternatively, before the transport operation, you can rename either the tablespace to be transported or the target tablespace.

  Starting with Oracle Database 12*c* Release 2 (12.2), the Recovery Manager (RMAN) `RECOVER` command can move tables to a different schema while remapping a tablespace. See *Oracle Database Backup and Recovery User's Guide* for more information.

- In a CDB, the default Data Pump directory object, `DATA_PUMP_DIR`, does not work with PDBs. You must define an explicit directory object within the PDB that you are using with Data Pump export/import.

- Transporting data with XMLTypes has the following limitations:

  – The target database must have XML DB installed.

  – Schemas referenced by XMLType tables cannot be the XML DB standard schemas.

  – If the schema for a transported XMLType table is not present in the target database, then it is imported and registered. If the schema already exists in the target database, then a message is displayed during import.

  – You must use only Data Pump to export and import the metadata for data that contains XMLTypes.

  The following query returns a list of tablespaces that contain XMLTypes:

  ```
  select distinct p.tablespace_name from dba_tablespaces p,
    dba_xml_tables x, dba_users u, all_all_tables t where
    t.table_name=x.table_name and t.tablespace_name=p.tablespace_name
    and x.owner=u.username;
  ```

  See *Oracle XML DB Developer's Guide* for information on XMLTypes.

- Types whose interpretation is application-specific and opaque to the database (such as `RAW`, `BFILE`, and the AnyTypes) can be transported, but they are not converted as part of the cross-platform transport operation. Their actual structure is known only to the application, so the application must address any endianness issues after these types are moved to the new platform. Types and objects that use these opaque types, either directly or indirectly, are also subject to this limitation.

- When you transport a tablespace containing tables with `TIMESTAMP WITH LOCAL TIME ZONE` (TSLTZ) data between databases with different time zones, the tables with the TSLTZ data are not transported. Error messages describe the tables that were not transported. However, tables in the tablespace that do not contain TSLTZ data are transported.

  You can determine the time zone of a database with the following query:

  ```
  SELECT DBTIMEZONE FROM DUAL;
  ```

  You can alter the time zone for a database with an `ALTER DATABASE` SQL statement.

You can use Data Pump to perform a conventional export/import of tables with TSLTZ data after the transport operation completes.

- Analytic workspaces cannot be part of cross-platform transport operations. If the source platform and target platform are different, then use Data Pump export/import to export and import analytic workspaces. See *Oracle OLAP DML Reference* for more information about analytic workspaces.

> **Note:**
>
> Do not invoke Data Pump export utility `expdp` or import utility `impdp` as `SYSDBA`, except at the request of Oracle technical support. `SYSDBA` is used internally and has specialized functions; its behavior is not the same as for general users.

Compatibility Considerations for Transporting Data

When transporting data, Oracle Database computes the lowest compatibility level at which the target database must run.

You can transport a tablespace or a table from a source database to a target database having the same or higher compatibility setting using transportable tablespaces, even if the target database is on the same or a different platform. The data transport operation fails if the compatibility level of the source database is higher than the compatibility level of the target database.

The following table shows the minimum compatibility requirements of the source and target databases in various scenarios. The source and target database need not have the same compatibility setting.

Table 1-1 Minimum Compatibility Requirements

Transport Scenario	Minimum Compatibility Setting	
	Source Database	**Target Database**
Transporting a database using full transportable export/import	12.0 (`COMPATIBLE` initialization parameter setting for an Oracle Database 12*c* or later database 12 (`VERSION` Data Pump export parameter setting for an 11.2.0.3 or later database)	12.0 (`COMPATIBLE` initialization parameter setting)
Transporting a tablespace between databases on the same platform using transportable tablespaces	8.0 (`COMPATIBLE` initialization parameter setting)	8.0 (`COMPATIBLE` initialization parameter setting)
Transporting a tablespace with different database block size than the target database using transportable tablespaces	9.0 (`COMPATIBLE` initialization parameter setting)	9.0 (`COMPATIBLE` initialization parameter setting)

Table 1-1 (Cont.) Minimum Compatibility Requirements

Transport Scenario	Minimum Compatibility Setting	
	Source Database	Target Database
Transporting a tablespace between databases on different platforms using transportable tablespaces	10.0 (`COMPATIBLE` initialization parameter setting)	10.0 (`COMPATIBLE` initialization parameter setting)
Transporting tables between databases	11.2.0 (`COMPATIBLE` initialization parameter setting for an Oracle Database 12*c* or later database	11.2.0 (`COMPATIBLE` initialization parameter setting)

Note:

- When you use full transportable export/import, the source database must be an Oracle Database 11*g* Release 2 (11.2.0.3) or later database, and the target database must be an Oracle Database 12*c* or later database.
- When transporting a database from Oracle Database 11*g* Release 2 (11.2.0.3) or later database to Oracle Database 12*c* or later database, you must set the Data Pump export parameter `VERSION` to `12` or higher.
- When transporting a database from an Oracle Database 19c database to an Oracle Database 19c database, you must set the initialization parameter `COMPATIBLE` to `19.0.0` or higher.

Limitations on Full Transportable Export/import

There are limitations on full transportable export/import.

Be aware of the following limitations on full transportable export/import:

- The general limitations described in "General Limitations on Transporting Data" apply to full transportable export/import.
- You cannot transport an encrypted tablespace to a platform with different endianness.

  To transport an encrypted tablespace to a platform with the same endianness, during export set the `ENCRYPTION_PWD_PROMPT` export utility parameter to `YES`, or use the `ENCRYPTION_PASSWORD` export utility parameter. During import, use the equivalent import utility parameter, and set the value to the same password that was used for the export.
- Full transportable export/import can export and import user-defined database objects in administrative tablespaces using conventional Data Pump export/import, such as direct path or external table. Administrative tablespaces are non-user tablespaces supplied with Oracle Database, such as the `SYSTEM` and `SYSAUX` tablespaces.

- Full transportable export/import cannot transport a database object that is defined in both an administrative tablespace (such as `SYSTEM` and `SYSAUX`) and a user-defined tablespace. For example, a partitioned table might be stored in both a user-defined tablespace and an administrative tablespace. If you have such database objects in your database, then you can redefine them before transporting them so that they are stored entirely in either an administrative tablespace or a user-defined tablespace. If the database objects cannot be redefined, then you can use conventional Data Pump export/import.
- When transporting a database over the network using full transportable export/import, auditing cannot be enabled for tables stored in an administrative tablespace (such as `SYSTEM` and `SYSAUX`) when the audit trail information itself is stored in a user-defined tablespace. See *Oracle Database Security Guide* for more information about auditing.

2

Converting Data to the Endian Format of the Target Operating System

Prepare for migrating your data by converting the endian format of the data to the endian format that is used on your target operating system platform.

- Converting Data Between Platforms
  When you perform a transportable operation, and the source platform and the target platform are of different endianness, you must convert the data being transported to the target platform format. If the source platform and the target platform are of the same endianness, then data conversion is not necessary. You can use the `DBMS_FILE_TRANSFER` package or the RMAN `CONVERT` command to convert data.
- Converting Data Between Platforms Using the DBMS_FILE_TRANSFER Package
  You can use the `GET_FILE` or `PUT_FILE` procedure of the `DBMS_FILE_TRANSFER` package to convert data between platforms during a data file transfer.
- Converting Data Between Platforms Using RMAN
  When you use the RMAN `CONVERT` command to convert data, you can either convert the data on the source platform after running Data Pump export, or you can convert the data on the target platform before running Data Pump import. In either case, you must transfer the data files from the source system to the target system.

Converting Data Between Platforms

When you perform a transportable operation, and the source platform and the target platform are of different endianness, you must convert the data being transported to the target platform format. If the source platform and the target platform are of the same endianness, then data conversion is not necessary. You can use the `DBMS_FILE_TRANSFER` package or the RMAN `CONVERT` command to convert data.

> **Note:**
>
> Some limitations might apply that are not described in these sections. Refer to the following documentation for more information:
>
> - "Transporting Data Across Platforms" for information about checking the endianness of platforms
> - *Oracle Database PL/SQL Packages and Types Reference* for information about limitations related to the `DBMS_FILE_TRANSFER` package
> - *Oracle Database Backup and Recovery Reference* for information about limitations related to the RMAN `CONVERT` command

Converting Data Between Platforms Using the DBMS_FILE_TRANSFER Package

You can use the `GET_FILE` or `PUT_FILE` procedure of the `DBMS_FILE_TRANSFER` package to convert data between platforms during a data file transfer.

When you use one of these procedures to move data files between the source platform and the target platform, each block in each data file is converted to the target platform's endianness.

This section uses an example to describe how to use the `DBMS_FILE_TRANSFER` package to convert a data file to a different platform. The example makes the following assumptions:

- The `GET_FILE` procedure will transfer the data file.
- The `mytable.342.123456789` data file is being transferred to a different platform.
- The endianness of the source platform is different from the endianness of the target platform.
- The global name of the source database is `dbsa.example.com`.
- Both the source database and the target database use Oracle Automatic Storage Management (Oracle ASM).

> **Note:**
>
> You can also use the `DBMS_FILE_TRANSFER` package to transfer data files between platforms with the same endianness.

Complete the following steps to convert the data file by transferring it with the `GET_FILE` procedure:

1. Use SQL*Plus to connect to the source database as an administrative user who can create directory objects.
2. Create a directory object to store the data files that you want to transfer to the target database.

   For example, to create a directory object named `sales_dir_source` for the `+data/dbsa/datafile` directory, execute the following SQL statement:

   ```
   CREATE OR REPLACE DIRECTORY sales_dir_source
      AS '+data/dbsa/datafile';
   ```

   The specified file system directory must exist when you create the directory object.
3. Use SQL*Plus to connect to the target database as an administrative user who can create database links, create directory objects, and run the procedures in the `DBMS_FILE_TRANSFER` package.
4. Create a database link from the target database to the source database.

   The connected user at the source database must have read privileges on the directory object that you created in Step 2.

5. Create a directory object to store the data files that you want to transfer from the source database.

   The user at the local database who will run the procedure in the `DBMS_FILE_TRANSFER` package must have write privileges on the directory object.

   For example, to create a directory object named `sales_dir_target` for the `+data/dbsb/datafile` directory, run the following SQL statement:

   ```
   CREATE OR REPLACE DIRECTORY sales_dir_target
     AS '+data/dbsb/datafile';
   ```

6. Run the `GET_FILE` procedure in the `DBMS_FILE_TRANSFER` package to transfer the data file.

   For example, run the following procedure to transfer the `mytable.342.123456789` data file from the source database to the target database using the database link you created in Step 4:

   ```
   BEGIN
     DBMS_FILE_TRANSFER.GET_FILE(
       source_directory_object      => 'sales_dir_source',
       source_file_name             => 'mytable.342.123456789',
       source_database              => 'dbsa.example.com',
       destination_directory_object => 'sales_dir_target',
       destination_file_name        => 'mytable');
   END;
   /
   ```

Note:

In this example, the destination data file name is `mytable`. Oracle ASM does not allow a fully qualified file name form in the `destination_file_name` parameter of the `GET_FILE` procedure.

See Also:

- *Oracle Database PL/SQL Packages and Types Reference* for more information about using the `DBMS_FILE_TRANSFER` package
- *Oracle Automatic Storage Management Administrator's Guide* for information about fully qualified file name forms in ASM
- *Oracle Database SQL Language Reference* for information about how to create a database link

Converting Data Between Platforms Using RMAN

When you use the RMAN `CONVERT` command to convert data, you can either convert the data on the source platform after running Data Pump export, or you can convert the data on the target platform before running Data Pump import. In either case, you must transfer the data files from the source system to the target system.

You can convert data with the following RMAN `CONVERT` commands:

- `CONVERT DATAFILE`
- `CONVERT TABLESPACE`
- `CONVERT DATABASE`

> **Note:**
>
> - Datatype restrictions apply to the RMAN `CONVERT` command.
> - RMAN `CONVERT` commands do not support conversion of data files between different endian formats for data files having undo segments.

- Converting Tablespaces on the Source System After Export
  An example illustrates how to use the RMAN `CONVERT TABLESPACE` command to convert tablespaces to a different platform.
- Converting Data Files on the Target System Before Import
  An example illustrates how to use the RMAN `CONVERT DATAFILE` command to convert data files to a different platform.

> **See Also:**
>
> - *Oracle Database Backup and Recovery Reference*
> - *Oracle Database Backup and Recovery User's Guide*

Converting Tablespaces on the Source System After Export

An example illustrates how to use the RMAN `CONVERT TABLESPACE` command to convert tablespaces to a different platform.

The example makes the following assumptions:

- The `sales_1` and `sales_2` tablespaces are being transported to a different platform.
- The endianness of the source platform is different from the endianness of the target platform.
- You want to convert the data on the source system, before transporting the tablespace set to the target system.
- You have completed the Data Pump export on the source database.

Complete the following steps to convert the tablespaces on the source system:

1. At a command prompt, start RMAN and connect to the source database:

```
$ RMAN TARGET /

Recovery Manager: Release 12.1.0.1.0 - Production
```

```
Copyright (c) 1982, 2012, Oracle and/or its affiliates.  All rights reserved.

connected to target database: salesdb (DBID=3295731590)
```

2. Use the RMAN `CONVERT TABLESPACE` command to convert the data files into a temporary location on the source platform.

   In this example, assume that the temporary location, directory `/tmp`, has already been created. The converted data files are assigned names by the system.

```
RMAN> CONVERT TABLESPACE sales_1,sales_2
2> TO PLATFORM 'Microsoft Windows IA (32-bit)'
3> FORMAT '/tmp/%U';

Starting conversion at source at 30-SEP-08
using channel ORA_DISK_1
channel ORA_DISK_1: starting datafile conversion
input datafile file number=00007 name=/u01/app/oracle/oradata/salesdb/
sales_101.dbf
converted datafile=/tmp/data_D-SALESDB_I-1192614013_TS-SALES_1_FNO-7_03jru08s
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:45
channel ORA_DISK_1: starting datafile conversion
input datafile file number=00008 name=/u01/app/oracle/oradata/salesdb/
sales_201.dbf
converted datafile=/tmp/data_D-SALESDB_I-1192614013_TS-SALES_2_FNO-8_04jru0aa
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:25
Finished conversion at source at 30-SEP-08
```

> **See Also:**
>
> *Oracle Database Backup and Recovery Reference* for a description of the RMAN `CONVERT` command

3. Exit Recovery Manager:

```
RMAN> exit
Recovery Manager complete.
```

4. Transfer the data files to the target system.

Converting Data Files on the Target System Before Import

An example illustrates how to use the RMAN `CONVERT DATAFILE` command to convert data files to a different platform.

During the conversion, you identify the data files by file name, not by tablespace name. Until the tablespace metadata is imported, the target instance has no way of knowing the desired tablespace names.

The example makes the following assumptions:

- You have not yet converted the data files for the tablespaces being transported.

  If you used the `DBMS_FILE_TRANSFER` package to transfer the data files to the target system, then the data files were converted automatically during the file transfer. See "Converting Data Between Platforms Using the DBMS_FILE_TRANSFER Package".

- The following data files are being transported to a different platform:
  - C:\Temp\sales_101.dbf
  - C:\Temp\sales_201.dbf
- The endianness of the source platform is different from the endianness of the target platform.
- You want to convert the data on the target system, before performing the Data Pump import.
- The converted data files are placed in C:\app\orauser\oradata\orawin\, which is the location of the existing data files for the target system:

Complete the following steps to convert the tablespaces on the target system:

1. If you are in SQL*Plus, then return to the host system:

```
SQL> HOST
```

2. Use the RMAN `CONVERT DATAFILE` command to convert the data files on the target platform:

```
C:\>RMAN TARGET /

Recovery Manager: Release 12.1.0.1.0 - Production

Copyright (c) 1982, 2012, Oracle and/or its affiliates.  All rights reserved.

connected to target database: ORAWIN (DBID=3462152886)

RMAN> CONVERT DATAFILE
2>'C:\Temp\sales_101.dbf',
3>'C:\Temp\sales_201.dbf'
4>TO PLATFORM="Microsoft Windows IA (32-bit)"
5>FROM PLATFORM="Solaris[tm] OE (32-bit)"
6>DB_FILE_NAME_CONVERT=
7>'C:\Temp\', 'C:\app\orauser\oradata\orawin\'
8> PARALLELISM=4;
```

   If the source location, the target location, or both do not use Oracle Automatic Storage Management (Oracle ASM), then the source and target platforms are optional. RMAN determines the source platform by examining the data file, and the target platform defaults to the platform of the host running the conversion.

   If both the source and target locations use Oracle ASM, then you must specify the source and target platforms in the `DB_FILE_NAME_CONVERT` clause.

> **See Also:**
>
> *Oracle Database Backup and Recovery Reference* for a description of the RMAN `CONVERT` command

3. Exit Recovery Manager:

```
RMAN> exit
Recovery Manager complete.
```

3

Connecting to the Database with SQL*Plus

Connect to the Oracle Database instance using SQL*Plus.

- Step 1: Open a Command Window
  Take the necessary action on your platform to open a window into which you can enter operating system commands.
- Step 2: Set Operating System Environment Variables
  Depending on your platform, you may have to set environment variables before starting SQL*Plus, or at least verify that they are set properly.
- Step 3: Start SQL*Plus
  Start SQL*Plus.
- Step 4: Submit the SQL*Plus CONNECT Command
  Submit the SQL*Plus `CONNECT` command to initially connect to the Oracle database instance or at any time to reconnect as a different user.

Step 1: Open a Command Window

Take the necessary action on your platform to open a window into which you can enter operating system commands.

- Open a command window.

Step 2: Set Operating System Environment Variables

Depending on your platform, you may have to set environment variables before starting SQL*Plus, or at least verify that they are set properly.

For example, on most platforms, you must set the environment variables `ORACLE_SID` and `ORACLE_HOME`. In addition, you must configure the `PATH` environment variable to include the *ORACLE_HOME*/bin directory. Some platforms may require additional environment variables:

- On UNIX and Linux, set environment variables by entering operating system commands.
- On Windows, Oracle Universal Installer (OUI) automatically assigns values to `ORACLE_HOME` and `ORACLE_SID` in the Windows registry.

If you did not create a database upon installation, OUI does not set `ORACLE_SID` in the registry; after you create your database at a later time, you must set the `ORACLE_SID` environment variable from a command window.

UNIX and Linux installations come with two scripts, `oraenv` and `coraenv`, that you can use to easily set environment variables. For more information, see *Oracle Database Administrator's Reference for Linux and UNIX-Based Operating Systems*.

For all platforms, when switching between instances with different Oracle homes, you must change the ORACLE_HOME environment variable. If multiple instances share the same Oracle home, then you must change only ORACLE_SID when switching instances.

Example 3-1 Setting Environment Variables in UNIX (C Shell)

```
setenv ORACLE_SID orcl
setenv ORACLE_HOME /u01/app/oracle/product/18.0.0/db_1
setenv LD_LIBRARY_PATH $ORACLE_HOME/lib:/usr/lib:/usr/dt/lib:/usr/openwin/lib:/usr/ccs/lib
```

Example 3-2 Setting Environment Variables in UNIX (Bash Shell)

```
export ORACLE_SID=orcl
export ORACLE_HOME=/u01/app/oracle/product/18.0.0/db_1
export LD_LIBRARY_PATH=$ORACLE_HOME/lib:/usr/lib:/usr/dt/lib:/usr/openwin/lib:/usr/ccs/lib
```

Example 3-3 Setting Environment Variables in Windows

```
SET ORACLE_SID=orawin2
```

Example 3-3 assumes that ORACLE_HOME and ORACLE_SID are set in the registry but that you want to override the registry value of ORACLE_SID to connect to a different instance.

On Windows, environment variable values that you set in a command prompt window override the values in the registry.

Step 3: Start SQL*Plus

Start SQL*Plus.

1. Do one of the following:
   - Ensure that the PATH environment variable contains *ORACLE_HOME*/bin.
   - Change directory to *ORACLE_HOME*/bin. Ensure that the PATH environment variable contains a dot (“.”).
2. Enter the following command (case-sensitive on UNIX and Linux):

   ```
   sqlplus /nolog
   ```

   You can also run the sqlplus command by specifying its complete path:

   ```
   ORACLE_HOME/bin/sqlplus /nolog
   ```

Step 4: Submit the SQL*Plus CONNECT Command

Submit the SQL*Plus CONNECT command to initially connect to the Oracle database instance or at any time to reconnect as a different user.

- In SQL*Plus, submit the CONNECT command.

Example 3-4 Connecting to a Local Database User

This simple example connects to a local database as user SYSTEM. SQL*Plus prompts for the SYSTEM user password.

```
connect system
```

Example 3-5 Connecting to a Local Database User with SYSDBA Privilege

This example connects to a local database as user SYS with the SYSDBA privilege. SQL*Plus prompts for the SYS user password.

```
connect sys as sysdba
```

When connecting as user SYS, you must connect AS SYSDBA.

Example 3-6 Connecting to a Local Database User with SYSBACKUP Privilege

This example connects to a local database as user SYSBACKUP with the SYSBACKUP privilege. SQL*Plus prompts for the SYSBACKUP user password.

```
connect sysbackup as sysbackup
```

When connecting as user SYSBACKUP, you must connect AS SYSBACKUP.

Example 3-7 Connecting Locally with SYSDBA Privilege with Operating System Authentication

This example connects locally with the SYSDBA privilege with operating system authentication.

```
connect / as sysdba
```

Example 3-8 Connecting with Easy Connect Syntax

This example uses easy connect syntax to connect as user salesadmin to a remote database running on the host dbhost.example.com. The Oracle Net listener (the listener) is listening on the default port (1521). The database service is sales.example.com. SQL*Plus prompts for the salesadmin user password.

```
connect salesadmin@"dbhost.example.com/sales.example.com"
```

Example 3-9 Connecting with Easy Connect Syntax with the Service Handler Type Indicated

This example is identical to Example 3-8, except that the service handler type is indicated.

```
connect salesadmin@"dbhost.example.com/sales.example.com:dedicated"
```

Example 3-10 Connecting with Easy Connect Syntax with a Nondefault Listener Port

This example is identical to Example 3-8, except that the listener is listening on the nondefault port number 1522.

```
connect salesadmin@"dbhost.example.com:1522/sales.example.com"
```

Example 3-11 Connecting with Easy Connect Syntax with the Host IP Address

This example is identical to Example 3-8, except that the host IP address is substituted for the host name.

```
connect salesadmin@"192.0.2.5/sales.example.com"
```

Example 3-12 Connecting with an IPv6 Address

This example connects using an IPv6 address. Note the enclosing square brackets.

```
connect salesadmin@"[2001:0DB8:0:0::200C:417A]/sales.example.com"
```

Example 3-13 Connecting by Specifying an Instance

This example specifies the instance to which to connect and omits the database service name. Note that when you specify only the instance, you cannot specify the service handler type.

```
connect salesadmin@"dbhost.example.com//orcl"
```

Example 3-14 Connecting with a Net Service Name

This example connects remotely as user `salesadmin` to the database service designated by the net service name `sales1`. SQL*Plus prompts for the `salesadmin` user password.

```
connect salesadmin@sales1
```

Example 3-15 Connecting with External Authentication

This example connects remotely with external authentication to the database service designated by the net service name `sales1`.

```
connect /@sales1
```

Example 3-16 Connecting with SYSDBA Privilege and External Authentication

This example connects remotely with the `SYSDBA` privilege and with external authentication to the database service designated by the net service name `sales1`.

```
connect /@sales1 as sysdba
```

Example 3-17 Connecting as a User with a Service Name

This example connects remotely as user `salesadmin` to the database service designated by the net service name `sales1`. The database session starts in the `rev21` edition. SQL*Plus prompts for the `salesadmin` user password.

```
connect salesadmin@sales1 edition=rev21
```

4
Migrating Oracle Database

Use the full transportable export/import feature to copy an entire database from one Oracle Database instance to another.

- Checking Character Set Compatibility
  Run these commands on the source and destination databases to find character sets that are compatible.
- Transporting a Database Using an Export Dump File
  You can transport a database using an export dump file.

Checking Character Set Compatibility

Run these commands on the source and destination databases to find character sets that are compatible.

```
SQL> show parameter CHARACTER

NAME                    TYPE        VALUE
---------               ---------   ----------
nls_numeric_characters     string


SQL> select * from database_properties where PROPERTY_NAME in
('NLS_CHARACTERSET','NLS_NCHAR_CHARACTERSET');

PROPERTY_NAME            PROPERTY_VALUE DESCRIPTION
-----------------------  ------------    ------------------
NLS_NCHAR_CHARACTERSET   AL16UTF16      NCHAR Character set
NLS_CHARACTERSET         WE8MSWIN1252   Character set


SQL> select * from nls_database_parameters where parameter like '%SET%';

PROPERTY_NAME              VALUE
------------------------   ---------
NLS_NCHAR_CHARACTERSET     AL16UTF16
NLS_CHARACTERSET           WE8MSWIN1252
```

Transporting a Database Using an Export Dump File

You can transport a database using an export dump file.

The following list of tasks summarizes the process of transporting a database using an export dump file. Details for each task are provided in the subsequent example.

1. At the source database, configure each of the user-defined tablespaces in read-only mode and export the database.

Ensure that the following parameters are set to the specified values:

- `TRANSPORTABLE=ALWAYS`
- `FULL=Y`

If the source database is an Oracle Database 11*g* database (11.2.0.3 or later), then you must set the `VERSION` parameter to `12` or higher.

If the source database contains any encrypted tablespaces or tablespaces containing tables with encrypted columns, then you must either specify `ENCRYPTION_PWD_PROMPT=YES`, or specify the `ENCRYPTION_PASSWORD` parameter.

The export dump file includes the metadata for objects contained within the user-defined tablespaces and both the metadata and data for user-defined objects contained within the administrative tablespaces, such as `SYSTEM` and `SYSAUX`.

2. Transport the export dump file.

   Copy the export dump file to a place that is accessible to the target database.

3. Transport the data files for all of the user-defined tablespaces in the database.

   Copy the data files to a place that is accessible to the target database.

   If the source platform and target platform are different, then check the endian format of each platform by running the query on the `V$TRANSPORTABLE_PLATFORM` view in "Transporting Data Across Platforms".

   If the source platform's endian format is different from the target platform's endian format, then use one of the following methods to convert the data files:

   - Use the `GET_FILE` or `PUT_FILE` procedure in the `DBMS_FILE_TRANSFER` package to transfer the data files. These procedures convert the data files to the target platform's endian format automatically.
   - Use the RMAN `CONVERT` command to convert the data files to the target platform's endian format.

   > **Note:**
   >
   > Conversion of data files between different endian formats is not supported for data files having undo segments.

   See "Converting Data Between Platforms" for more information.

4. (Optional) Restore the user-defined tablespaces to read/write mode on the source database.

5. At the target database, import the database.

   When the import is complete, the user-defined tablespaces are in read/write mode.

Example

The tasks for transporting a database are illustrated in detail in this example. This example assumes that the source platform is Solaris and the target platform is Microsoft Windows.

It also assumes that the source platform has the following data files and tablespaces:

Tablespace	Type	Data File
sales	User-defined	/u01/app/oracle/oradata/mydb/sales01.dbf
customers	User-defined	/u01/app/oracle/oradata/mydb/cust01.dbf
employees	User-defined	/u01/app/oracle/oradata/mydb/emp01.dbf
SYSTEM	Administrative	/u01/app/oracle/oradata/mydb/system01.dbf
SYSAUX	Administrative	/u01/app/oracle/oradata/mydb/sysaux01.dbf

This example makes the following additional assumptions:

- The target database is a new database that is being populated with the data from the source database. The name of the source database is `mydb`.
- Both the source database and the target database are Oracle Database 19c databases.

Complete the following tasks to transport the database using an export dump file:

Task 1 Generate the Export Dump File
Generate the export dump file by completing the following steps:

1. Start SQL*Plus and connect to the database as an administrator or as a user who has either the `ALTER TABLESPACE` or `MANAGE TABLESPACE` system privilege.
2. Make all of the user-defined tablespaces in the database read-only.

```
ALTER TABLESPACE sales READ ONLY;

ALTER TABLESPACE customers READ ONLY;

ALTER TABLESPACE employees READ ONLY;
```

3. Invoke the Data Pump export utility as a user with `DATAPUMP_EXP_FULL_DATABASE` role and specify the full transportable export/import options.

```
SQL> HOST

$ expdp user_name full=y dumpfile=expdat.dmp directory=data_pump_dir
        transportable=always logfile=export.log

Password: password
```

   You must always specify `TRANSPORTABLE=ALWAYS`, which determines whether the transportable option is used.

   This example specifies the following Data Pump parameters:

   - The `FULL` parameter specifies that the entire database is being exported.
   - The `DUMPFILE` parameter specifies the name of the structural information export dump file to be created, `expdat.dmp`.
   - The `DIRECTORY` parameter specifies the directory object that points to the operating system or Oracle Automatic Storage Management location of the dump file. You must create the `DIRECTORY` object before invoking Data Pump, and you must grant the `READ` and `WRITE` object privileges on the directory to

the user running the Export utility. See *Oracle Database SQL Language Reference* for information on the `CREATE DIRECTORY` command.

In a non-CDB, the directory object `DATA_PUMP_DIR` is created automatically. Read and write access to this directory is automatically granted to the `DBA` role, and thus to users `SYS` and `SYSTEM`.

However, the directory object `DATA_PUMP_DIR` is not created automatically in a PDB. Therefore, when importing into a PDB, create a directory object in the PDB and specify the directory object when you run Data Pump.

> **See Also:**
>
> - *Oracle Database Utilities* for information about the default directory when the `DIRECTORY` parameter is omitted
> - *Oracle Multitenant Administrator's Guide* for more information about PDBs

- The `LOGFILE` parameter specifies the file name of the log file to be written by the export utility. In this example, the log file is written to the same directory as the dump file, but it can be written to a different location.

To perform a full transportable export on an Oracle Database 11*g* Release 2 (11.2.0.3) or later Oracle Database 11*g* database, use the `VERSION` parameter, as shown in the following example:

```
expdp user_name full=y dumpfile=expdat.dmp directory=data_pump_dir
      transportable=always version=12 logfile=export.log
```

Full transportable import is supported only for Oracle Database 12*c* and later databases.

> **Note:**
>
> In this example, the Data Pump utility is used to export only data dictionary structural information (metadata) for the user-defined tablespaces. Actual data is unloaded only for the administrative tablespaces (`SYSTEM` and `SYSAUX`), so this operation goes relatively quickly even for large user-defined tablespaces.

4. Check the log file for errors, and take note of the dump file and data files that you must transport to the target database. `expdp` outputs the names and paths of these files in messages like these:

```
******************************************************************************
Dump file set for SYSTEM.SYS_EXPORT_TRANSPORTABLE_01 is:
  /u01/app/oracle/admin/mydb/dpdump/expdat.dmp
******************************************************************************
Datafiles required for transportable tablespace SALES:
  /u01/app/oracle/oradata/mydb/sales01.dbf
Datafiles required for transportable tablespace CUSTOMERS:
```

```
  /u01/app/oracle/oradata/mydb/cust01.dbf
Datafiles required for transportable tablespace EMPLOYEES:
  /u01/app/oracle/oradata/mydb/emp01.dbf
```

5. When finished, exit back to SQL*Plus:

```
$ exit
```

> **See Also:**
>
> *Oracle Database Utilities* for information about using the Data Pump utility

Task 2 Transport the Export Dump File

Transport the dump file to the directory pointed to by the DATA_PUMP_DIR directory object, or to any other directory of your choosing. The new location must be accessible to the target database.

At the target database, run the following query to determine the location of DATA_PUMP_DIR:

```
SELECT * FROM DBA_DIRECTORIES WHERE DIRECTORY_NAME = 'DATA_PUMP_DIR';

OWNER      DIRECTORY_NAME   DIRECTORY_PATH
---------- ---------------- ----------------------------------
SYS        DATA_PUMP_DIR    C:\app\orauser\admin\orawin\dpdump\
```

Task 3 Transport the Data Files for the User-Defined Tablespaces

Transport the data files of the user-defined tablespaces in the database to a place that is accessible to the target database.

In this example, transfer the following data files from the source database to the target database:

- sales01.dbf
- cust01.dbf
- emp01.dbf

If you are transporting the database to a platform different from the source platform, then determine if cross-platform database transport is supported for both the source and target platforms, and determine the endianness of each platform. If both platforms have the same endianness, then no conversion is necessary. Otherwise, you must do a conversion of each tablespace in the database, either at the source database or at the target database.

If you are transporting the database to a different platform, you can execute the following query on each platform. If the query returns a row, then the platform supports cross-platform tablespace transport.

```
SELECT d.PLATFORM_NAME, ENDIAN_FORMAT
     FROM V$TRANSPORTABLE_PLATFORM tp, V$DATABASE d
     WHERE tp.PLATFORM_NAME = d.PLATFORM_NAME;
```

The following is the query result from the source platform:

```
PLATFORM_NAME                    ENDIAN_FORMAT
-------------------------------- --------------
Solaris[tm] OE (32-bit)          Big
```

The following is the query result from the target platform:

```
PLATFORM_NAME                    ENDIAN_FORMAT
-------------------------------- --------------
Microsoft Windows IA (32-bit)    Little
```

In this example, you can see that the endian formats are different. Therefore, in this case, a conversion is necessary for transporting the database. Use either the `GET_FILE` or `PUT_FILE` procedure in the `DBMS_FILE_TRANSFER` package to transfer the data files. These procedures convert the data files to the target platform's endian format automatically. Transport the data files to the location of the existing data files of the target database. On the UNIX and Linux platforms, this location is typically /u01/app/oracle/oradata/*dbname*/ or *+DISKGROUP*/*dbname*/datafile/. Alternatively, you can use the RMAN `CONVERT` command to convert the data files. See "Converting Data Between Platforms" for more information.

> **Note:**
>
> If no endianness conversion of the tablespaces is needed, then you can transfer the files using any file transfer method.

Task 4 (Optional) Restore Tablespaces to Read/Write Mode
Make the transported tablespaces read/write again at the source database, as follows:

```
ALTER TABLESPACE sales READ WRITE;
ALTER TABLESPACE customers READ WRITE;
ALTER TABLESPACE employees READ WRITE;
```

You can postpone this task to first ensure that the import process succeeds.

Task 5 At the Target Database, Import the Database
Invoke the Data Pump import utility as a user with `DATAPUMP_IMP_FULL_DATABASE` role and specify the full transportable export/import options.

```
impdp user_name full=Y dumpfile=expdat.dmp directory=data_pump_dir
   transport_datafiles=
      '/u01/app/oracle/oradata/mydb/sales01.dbf',
      '/u01/app/oracle/oradata/mydb/cust01.dbf',
      '/u01/app/oracle/oradata/mydb/emp01.dbf'
   logfile=import.log

Password: password
```

This example specifies the following Data Pump parameters:

- The `FULL` parameter specifies that the entire database is being imported in `FULL` mode.
- The `DUMPFILE` parameter specifies the exported file containing the metadata for the user-defined tablespaces and both the metadata and data for the administrative tablespaces to be imported.
- The `DIRECTORY` parameter specifies the directory object that identifies the location of the export dump file. You must create the `DIRECTORY` object before invoking Data Pump, and you must grant the `READ` and `WRITE` object privileges on the

directory to the user running the Import utility. See *Oracle Database SQL Language Reference* for information on the `CREATE DIRECTORY` command.

In a non-CDB, the directory object `DATA_PUMP_DIR` is created automatically. Read and write access to this directory is automatically granted to the `DBA` role, and thus to users `SYS` and `SYSTEM`.

However, the directory object `DATA_PUMP_DIR` is not created automatically in a PDB. Therefore, when importing into a PDB, create a directory object in the PDB and specify the directory object when you run Data Pump.

> **See Also:**
>
> - *Oracle Database Utilities* for information about the default directory when the `DIRECTORY` parameter is omitted
> - *Oracle Multitenant Administrator's Guide* for more information about PDBs

- The `TRANSPORT_DATAFILES` parameter identifies all of the data files to be imported.

  You can specify the `TRANSPORT_DATAFILES` parameter multiple times in a parameter file specified with the `PARFILE` parameter if there are many data files.

- The `LOGFILE` parameter specifies the file name of the log file to be written by the import utility. In this example, the log file is written to the directory from which the dump file is read, but it can be written to a different location.

After this statement executes successfully, check the import log file to ensure that no unexpected error has occurred.
When dealing with a large number of data files, specifying the list of data file names in the statement line can be a laborious process. It can even exceed the statement line limit. In this situation, you can use an import parameter file. For example, you can invoke the Data Pump import utility as follows:

```
impdp user_name parfile='par.f'
```

For example, `par.f` might contain the following lines:

```
FULL=Y
DUMPFILE=expdat.dmp
DIRECTORY=data_pump_dir
TRANSPORT_DATAFILES=
'/u01/app/oracle/oradata/mydb/sales01.dbf',
'/u01/app/oracle/oradata/mydb/cust01.dbf',
'/u01/app/oracle/oradata/mydb/emp01.dbf'
LOGFILE=import.log
```

Note:

- During the import, user-defined tablespaces might be temporarily made read/write for metadata loading. Ensure that no user changes are made to the data during the import. At the successful completion of the import, all user-defined tablespaces are made read/write.
- When performing a network database import, the `TRANSPORTABLE` parameter must be set to `always`.
- When you are importing into a PDB in a CDB, specify the connect identifier for the PDB after the user name. For example, if the connect identifier for the PDB is `hrpdb`, then enter the following when you run the Oracle Data Pump Import utility:

  ```
  impdp user_name@hrpdb ...
  ```

See Also:

- *Oracle Database Utilities* for information about using the import utility
- *Oracle Multitenant Administrator's Guide*

Index

C

CONNECT command, SQL*Plus, *3-2*

E

editions
  in CONNECT command, *3-2*

F

full transportable export/import, *1-1*

I

import operations
  PDBs, *4-8*

P

PDBs
  import operations, *4-8*

T

tablespaces
  containing XMLTypes, *1-3*
transporting data
  across platforms, *1-1*
  character sets, *1-3*
  compatibility considerations, *1-5*
  full transportable export/import, *1-1*
    limitations, *1-6*
  limitations, *1-3*
  national character sets, *1-3*
  PDBs, *4-8*
  XMLTypes in, *1-3*

X

XMLTypes
  transporting data, *1-3*