Oracle® Database

Migrating Non-CDBs to New Hardware with a Different Endian Operating System and for the Same Release

18c
E97283-02
September 2018

Oracle Database Migrating Non-CDBs to New Hardware with a Different Endian Operating System and for the Same Release, 18c

E97283-02

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 v

1 Migrating Data Using Transportable Tablespaces

Transporting Data Across Platforms 1-1
General Limitations on Transporting Data 1-3
Compatibility Considerations for Transporting Data 1-5
Limitations on Transportable Tablespaces 1-6

2 Preparing the Target Server

Creating the Target Database 2-1

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

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

4 Connecting to the Database with SQL*Plus

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

5 Migrating Oracle Database

Checking Character Set Compatibility 5-1
Transporting Tablespaces Between Databases 5-2
Task 1: Pick a Self-Contained Set of Tablespaces 5-3
Task 2: Generate a Transportable Tablespace Set 5-5
Task 3: Transport the Export Dump File 5-7
Task 4: Transport the Tablespace Set 5-7
Task 5: (Optional) Restore Tablespaces to Read/Write Mode 5-8
Task 6: Import the Tablespace Set 5-8
Postmigration Tasks 5-10

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 migrate the same release Oracle Database to a different endian operating system using transportable tablespaces with RMAN.

Prerequisites for this Scenario

- Ensure that the source and target databases have identical character sets, or at least that the destination database character set is a superset of the source database.
- Before you run Oracle Data Pump import on the destination server, run the `RMAN CONVERT` command to convert data.
- Run the migration procedures as the `oracle` user.

Outline for this Scenario

1. **Migrating Data Using Transportable Tablespaces**. Migrate data in preparation for the database migration.
2. **Preparing the Target Server**. Create a target database in preparation for the migration.
3. **Converting Data to the Endian Format of the Target Operating System**. Convert your tablespaces in preparation for the migration.
4. **Connecting to the Oracle Database with SQL*Plus**.
5. **Migrating Oracle Database**. Migrate your database using transportable tablespaces, and complete postmigration tasks.

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

Migrating Data Using Transportable Tablespaces

Transportable tablespaces and transportable tables transport data that resides in user-defined tablespaces.

Use the transportable tablespaces feature to move a set of tablespaces between databases.

- 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 Transportable Tablespaces
  This section lists the limitations on transportable tablespace.

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)

Table 1-1 (Cont.) Minimum Compatibility Requirements

Transport Scenario	Minimum Compatibility Setting	
	Source Database	Target Database
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)
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)

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 18c database to an Oracle Database 18c database, you must set the initialization parameter `COMPATIBLE` to `18.0.0` or higher.

Limitations on Transportable Tablespaces

This section lists the limitations on transportable tablespace.

Be aware of the following limitations for transportable tablespaces:

- The general limitations described in "General Limitations on Transporting Data" apply to transportable tablespaces.
- When transporting a tablespace set, objects with underlying objects (such as materialized views) or contained objects (such as partitioned tables) are not transportable unless all of the underlying or contained objects are in the tablespace set.
- Transportable tablespaces cannot transport tables with `TIMESTAMP WITH TIMEZONE` (TSTZ) data across platforms with different time zone file versions. The transportable tablespace operation skips these tables. You can export and import these tables conventionally.

  See *Oracle Database Utilities* for more information.
- You cannot include administrative tablespaces, such as `SYSTEM` and `SYSAUX` in a transportable tablespace set.

2
Preparing the Target Server

This chapter shows how to use Database Configuration Assistant (DBCA) to create the destination database in the target server.

Run the PING command on source and target server to check if both of the servers are reachable.

- Creating the Target Database
  Create a new database on the target system.

Creating the Target Database

Create a new database on the target system.

The new target database consists initially of just `SYSTEM`, `SYSAUX`, `UNDO`, temporary tablespaces, and user tablespaces. You can use DBCA to create the target database.

When creating the target database, consider the following:

- Although all user tablespaces from the original source database are transported and plugged into the new target database, initially the target database must contain placeholder tablespaces for the user tablespaces that will be transported. The size of the user tablespaces initially created in the target database can be small; the target tablespaces do not have to match the sizes in the source database. The placeholder tablespaces will be dropped from the target database before transporting the tablespaces from the source system.
- The sizes of the `SYSTEM`, `SYSAUX`, `UNDO`, and temporary tablespaces must be same or larger than those tablespaces on the source database.
- The sizes of log files and number of members for each log file group in the new target database should be the same as, or larger than, the source database.
- The source and target database must use the same character set and same national character set. Check the source database character sets by issuing the following query:

  ```
  SQL> select * from database_properties where property_name like '%CHARACTERSET';
  ```

- The database options and components used in the source database should be installed on the target database.
  - Query the `V$OPTION` view to get currently installed database options.
  - Query `DBA_REGISTRY` to get currently installed database components.

3

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.
```

4

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 4-1 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 4-2 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 4-3 Setting Environment Variables in Windows

```
SET ORACLE_SID=orawin2
```

Example 4-3 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 4-4 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 4-5 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 4-6 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 4-7 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 4-8 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 4-9 Connecting with Easy Connect Syntax with the Service Handler Type Indicated

This example is identical to Example 4-8, except that the service handler type is indicated.

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

Example 4-10 Connecting with Easy Connect Syntax with a Nondefault Listener Port

This example is identical to Example 4-8, except that the listener is listening on the nondefault port number 1522.

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

Example 4-11 Connecting with Easy Connect Syntax with the Host IP Address

This example is identical to Example 4-8, except that the host IP address is substituted for the host name.

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

Example 4-12 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 4-13 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 4-14 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 4-15 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 4-16 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 4-17 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
```

5
Migrating Oracle Database

Use the transportable tablespaces feature to copy a set of tablespaces from one Oracle Database to another.

- Checking Character Set Compatibility
  Run these commands on the source and destination databases to find character sets that are compatible.
- Transporting Tablespaces Between Databases
  You can transport a tablespace or a set of tablespaces between databases.
- Task 1: Pick a Self-Contained Set of Tablespaces
  There may be logical or physical dependencies between the database objects in the transportable set and the database objects outside of the transportable set. You can only transport a tablespace set that is self-contained, that is, none of the database objects inside a tablespace set are dependent on any of the database objects outside of that tablespace set.
- Task 2: Generate a Transportable Tablespace Set
  After ensuring that you have a self-contained set of tablespaces that you want to transport, generate a transportable tablespace set.
- Task 3: 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.
- Task 4: Transport the Tablespace Set
  Transport the data files of the tablespaces to a directory that is accessible to the target database.
- Task 5: (Optional) Restore Tablespaces to Read/Write Mode
  Make the transported tablespaces read/write again at the source database.
- Task 6: Import the Tablespace Set
  To complete the transportable tablespaces operation, import the tablespace set.
- Postmigration Tasks
  Complete these tasks to prepare the target Oracle Database for use.

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 Tablespaces Between Databases

You can transport a tablespace or a set of tablespaces between databases.

The following list of tasks summarizes the process of transporting a tablespace. Details for each task are provided in the subsequent example.

1. Pick a self-contained set of tablespaces.
2. At the source database, configure the set of tablespaces in read-only mode and generate a transportable tablespace set.

   A transportable tablespace set (or transportable set) consists of data files for the set of tablespaces being transported and an export dump file containing structural information (metadata) for the set of tablespaces. You use Data Pump to perform the export.
3. Transport the export dump file.

   Copy the export dump file to a place that is accessible to the target database.
4. Transport the tablespace set.

   Copy the data files to a directory 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.

   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.
5. (Optional) Restore tablespaces to read/write mode on the source database.
6. At the target database, import the tablespace set.

Run the Data Pump utility to import the metadata for the tablespace set.

Example 5-1 Example

These tasks for transporting a tablespace are illustrated more fully in the example that follows, where it is assumed the following data files and tablespaces exist:

Tablespace	Data File
sales_1	/u01/app/oracle/oradata/salesdb/sales_101.dbf
sales_2	/u01/app/oracle/oradata/salesdb/sales_201.dbf

Task 1: Pick a Self-Contained Set of Tablespaces

There may be logical or physical dependencies between the database objects in the transportable set and the database objects outside of the transportable set. You can only transport a tablespace set that is self-contained, that is, none of the database objects inside a tablespace set are dependent on any of the database objects outside of that tablespace set.

Some examples of self-contained tablespace violations are:

- An index inside the set of tablespaces is for a table outside of the set of tablespaces.

  > **Note:**
  >
  > It is not a violation if a corresponding index for a table is outside of the set of tablespaces.

- A partitioned table is partially contained in the set of tablespaces.

  The tablespace set that you want to copy must contain either all partitions of a partitioned table, or none of the partitions of a partitioned table. To transport a subset of a partition table, you must exchange the partitions into tables.

  See *Oracle Database VLDB and Partitioning Guide* for information about exchanging partitions.

- A referential integrity constraint points to a table across a set boundary.

  When transporting a set of tablespaces, you can choose to include referential integrity constraints. However, doing so can affect whether a set of tablespaces is self-contained. If you decide not to transport constraints, then the constraints are not considered as pointers.

- A table inside the set of tablespaces contains a `LOB` column that points to `LOBS` outside the set of tablespaces.

- An XML DB schema (*.xsd) that was registered by user A imports a global schema that was registered by user B, and the following is true: the default tablespace for user A is tablespace A, the default tablespace for user B is tablespace B, and only tablespace A is included in the set of tablespaces.

To determine whether a set of tablespaces is self-contained, run the `TRANSPORT_SET_CHECK` procedure in the Oracle supplied package `DBMS_TTS`. You must

have been granted the EXECUTE_CATALOG_ROLE role (initially signed to SYS) to run this procedure.

When you run the DBMS_TTS.TRANSPORT_SET_CHECK procedure, specify the list of tablespaces in the transportable set to be checked for self containment. You can optionally specify if constraints must be included. For strict or full containment, you must additionally set the TTS_FULL_CHECK parameter to TRUE.

The strict or full containment check is for cases that require capturing not only references going outside the transportable set, but also those coming into the set. Tablespace Point-in-Time Recovery (TSPITR) is one such case where dependent objects must be fully contained or fully outside the transportable set.

For example, it is a violation to perform TSPITR on a tablespace containing a table t but not its index i because the index and data will be inconsistent after the transport. A full containment check ensures that there are no dependencies going outside or coming into the transportable set. See the example for TSPITR in the *Oracle Database Backup and Recovery User's Guide*.

> **Note:**
>
> The default for transportable tablespaces is to check for self containment rather than full containment.

The following statement can be used to determine whether tablespaces sales_1 and sales_2 are self-contained, with referential integrity constraints taken into consideration (indicated by TRUE).

```
EXECUTE DBMS_TTS.TRANSPORT_SET_CHECK('sales_1,sales_2', TRUE);
```

After running the DBMS_TTS.TRANSPORT_SET_CHECK procedure, you can see all the violations by selecting from the TRANSPORT_SET_VIOLATIONS view. If the set of tablespaces is self-contained, then this view is empty. The following example illustrates a case where there are two violations: a foreign key constraint, dept_fk, across the tablespace set boundary, and a partitioned table, jim.sales, that is partially contained in the tablespace set.

```
SELECT * FROM TRANSPORT_SET_VIOLATIONS;

VIOLATIONS
--------------------------------------------------------------------------------
Constraint DEPT_FK between table JIM.EMP in tablespace SALES_1 and table
JIM.DEPT in tablespace OTHER
Partitioned table JIM.SALES is partially contained in the transportable set
```

You must resolve these violations before sales_1 and sales_2 are transportable. As noted in the next task, one choice for bypassing the integrity constraint violation is to not to export the integrity constraints.

> **See Also:**
>
> - *Oracle Database PL/SQL Packages and Types Reference* for more information about the `DBMS_TTS` package
> - *Oracle Database Backup and Recovery User's Guide* for information specific to using the `DBMS_TTS` package for TSPITR

Task 2: Generate a Transportable Tablespace Set

After ensuring that you have a self-contained set of tablespaces that you want to transport, generate a transportable tablespace set.

To generate a transportable tablespace set:

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 tablespaces in the set read-only.

   ```
   ALTER TABLESPACE sales_1 READ ONLY;

   ALTER TABLESPACE sales_2 READ ONLY;
   ```

3. Run the Data Pump export utility as a user with `DATAPUMP_EXP_FULL_DATABASE` role and specify the tablespaces in the transportable set.

   ```
   SQL> HOST

   $ expdp user_name dumpfile=expdat.dmp directory=data_pump_dir
           transport_tablespaces=sales_1,sales_2 logfile=tts_export.log

   Password: password
   ```

   You must always specify `TRANSPORT_TABLESPACES`, which specifies that the transportable option is used. This example specifies the following additional Data Pump parameters:

   - 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.

     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.
   - The `LOGFILE` parameter specifies the log file to create for the export utility. In this example, the log file is created in the same directory as the dump file, but you can specify any other directory for storing the log file.

- Triggers and indexes are included in the export operation by default.

To perform a transport tablespace operation with a strict containment check, use the `TRANSPORT_FULL_CHECK` parameter, as shown in the following example:

```
expdp use_name dumpfile=expdat.dmp directory=data_pump_dir
      transport_tablespaces=sales_1,sales_2 transport_full_check=y
      logfile=tts_export.log
```

In this case, the Data Pump export utility verifies that there are no dependencies between the objects inside the transportable set and objects outside the transportable set. If the tablespace set being transported is not self-contained, then the export fails and indicates that the transportable set is not self-contained. You must resolve these violations and then run this task again.

> **Note:**
>
> In this example, the Data Pump utility is used to export only data dictionary structural information (metadata) for the tablespaces. No actual data is unloaded, so this operation goes relatively quickly even for large tablespace sets.

4. The `expdp` utility displays the names and paths of the dump file and the data files on the command line as shown in the following example. These are the files that you need to transport to the target database. Also, check the log file for any errors.

```
******************************************************************************
Dump file set for SYSTEM.SYS_EXPORT_TRANSPORTABLE_01 is:
  /u01/app/oracle/admin/salesdb/dpdump/expdat.dmp
******************************************************************************
Datafiles required for transportable tablespace SALES_1:
  /u01/app/oracle/oradata/salesdb/sales_101.dbf
Datafiles required for transportable tablespace SALES_2:
  /u01/app/oracle/oradata/salesdb/sales_201.dbf
```

5. When the Data Pump export operation is completed, exit the `expdp` utility to return to SQL*Plus:

```
$ EXIT
```

> **See Also:**
>
> - *Oracle Database SQL Language Reference* for information on the `CREATE DIRECTORY` command
> - *Oracle Database Utilities* for information about the default directory when the `DIRECTORY` parameter is omitted
> - *Oracle Database Utilities* for information about using the Data Pump utility
> - *Oracle Multitenant Administrator's Guide* for more information about PDBs

Task 3: 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 4: Transport the Tablespace Set

Transport the data files of the tablespaces to a directory that is accessible to the target database.

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

- sales_101.dbf
- sales_201.dbf

If you are transporting the tablespace set to a platform different from the source platform, then determine if cross-platform tablespace 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 the data conversion either at the source database or at the target database.

If you are transporting sales_1 and sales_2 to a different platform, then you can run the following query on each platform. If the query returns a row, 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 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 to convert the data files.

> **Note:**
>
> - If you use the RMAN `CONVERT` command, then conversion of data files between different endian formats is not supported for data files having undo segments.
> - If no endianness conversion of the tablespaces is needed, then you can transfer the files using any file transfer method.

Task 5: (Optional) Restore Tablespaces to Read/Write Mode

Make the transported tablespaces read/write again at the source database.

The following statements make the `sales_1` and `sales_2` tablespaces read/write:

```
ALTER TABLESPACE sales_1 READ WRITE;
ALTER TABLESPACE sales_2 READ WRITE;
```

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

Task 6: Import the Tablespace Set

To complete the transportable tablespaces operation, import the tablespace set.

To import the tablespace set:

1. Run the Data Pump import utility as a user with `DATAPUMP_IMP_FULL_DATABASE` role and import the tablespace metadata.

   ```
   impdp user_name dumpfile=expdat.dmp directory=data_pump_dir
      transport_datafiles=
      'c:\app\orauser\oradata\orawin\sales_101.dbf',
      'c:\app\orauser\oradata\orawin\sales_201.dbf'
      remap_schema=sales1:crm1  remap_schema=sales2:crm2
      logfile=tts_import.log

   Password: password
   ```

   This example specifies the following Data Pump parameters:

   - The `DUMPFILE` parameter specifies the exported file containing the metadata for the 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 running 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 database creates the directory object `DATA_PUMP_DIR` automatically. Read and write access to this directory is automatically granted to the `DBA` role, and thus to users `SYS` and `SYSTEM`.

However, the database does not create the directory object `DATA_PUMP_DIR` 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 containing the tablespaces 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 `REMAP_SCHEMA` parameter changes the ownership of database objects. If you do not specify `REMAP_SCHEMA`, then all database objects (such as tables and indexes) are created in the same user schema as in the source database, and those users must already exist in the target database. If they do not exist, then the import utility returns an error. In this example, objects in the tablespace set owned by `sales1` in the source database will be owned by `crm1` in the target database after the tablespace set is imported. Similarly, objects owned by `sales2` in the source database will be owned by `crm2` in the target database. In this case, the target database is not required to have users `sales1` and `sales2`, but must have users `crm1` and `crm2`.

  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 table. See *Oracle Database Backup and Recovery User's Guide* for more information.

- 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 runs successfully, all tablespaces in the set being copied remain in read-only mode. Check the import log file to ensure that no 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 as the data file list can even exceed the statement line limit. In this situation, you can use an import parameter file. For example, you can run the Data Pump import utility as follows:

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

The `par.f` parameter file contains the following:

```
DUMPFILE=expdat.dmp
DIRECTORY=data_pump_dir
TRANSPORT_DATAFILES=
'C:\app\orauser\oradata\orawin\sales_101.dbf',
'C:\app\orauser\oradata\orawin\sales_201.dbf'
REMAP_SCHEMA=sales1:crm1  REMAP_SCHEMA=sales2:crm2
LOGFILE=tts_import.log
```

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

2. If required, put the tablespaces into read/write mode on the target database.

Postmigration Tasks

Complete these tasks to prepare the target Oracle Database for use.

1. Verify if the data has imported to the target database.

   Run the following queries on the source and target databases to check if data was exported and imported completely without any errors.

   To view all users that exist in the database:

   ```
   SQL> SELECT count(*) FROM dba_users;
   SQL> SELECT username, account_status FROM dba_users;
   ```

   To view the total number of objects in the database:

   ```
   SQL> SELECT count(*) FROM dba_objects;
   SQL> SELECT count(*), owner FROM dba_objects group by owner;
   ```

   To view a list of all the tables owned by the current user:

   ```
   SQL> SELECT count(*) FROM user_tables;
   SQL> SELECT count(*), tablespace_name FROM user_tables group by tablespace_name;
   ```

   To view the exact size in MBytes occupied by the object at the tablespace:

   ```
   SELECT owner, segment_name, segment_type, partition_name, ROUND(bytes/
   (1024*1024),2) SIZE_MB, tablespace_name
   FROM DBA_SEGMENTS
   WHERE SEGMENT_TYPE IN ('TABLE', 'TABLE PARTITION', 'TABLE SUBPARTITION',
   'INDEX', 'INDEX PARTITION', 'INDEX SUBPARTITION', 'TEMPORARY', 'LOBINDEX',
   'LOBSEGMENT', 'LOB PARTITION')
   --AND TABLESPACE_NAME LIKE 'COSTE%'
   --AND SEGMENT_NAME LIKE 'P2010201%'
   --AND partition_name LIKE 'P20100201%'
   --AND segment_type = 'TABLE'
   --AND OWNER = 'TARGET_POC'
   --AND ROUND(bytes/(1024*1024),2) > 1000
   ORDER BY bytes DESC;
   ```

   To view the total space occupied in MBytes:

   ```
   SELECT tablespace_name, owner, segment_type "Object Type",
          COUNT(owner) "Number of Objects",
          ROUND(SUM(bytes) / 1024 / 1024, 2) "Total Size in MB"
   ```

```
FROM   sys.dba_segments
WHERE  tablespace_name IN ('MPIS')
GROUP BY tablespace_name, owner, segment_type
ORDER BY tablespace_name, owner, segment_type;
```

To view the size of the database:

```
SQL> SELECT a.data_size+b.temp_size+c.redo_size+d.controlfile_size "total_size
in MB" FROM ( select
    sum(bytes)/1024/1024 data_size
    FROM dba_data_files ) a,
    (select nvl(sum(bytes),0)/1024/1024 temp_size
    FROM dba_temp_files) b,
    (select sum(bytes)/1024/1024 redo_size
    FROM sys.v_$log) c,
    (select sum(BLOCK_SIZE*FILE_SIZE_BLKS)/1024/1024 controlfile_size from
v$controlfile) d;
```

2. Switch transported tablespaces to `READ WRITE` mode at destination.

   ```
   SQL> ALTER TABLESPACE tablespace name READ WRITE;
   ```

3. Revert tablespaces to `READ WRITE` mode at source.

   ```
   SQL> ALTER TABLESPACE tablespace name READ WRITE;
   ```

4. Redirect applications to destination database.

   Create and start appropriate database services and/or network connectivity on the new destination database.

5. Clean up the staging directories.

   Remove unneeded files from the source and destination hosts.

Index

C

CONNECT command, SQL*Plus, *4-2*

E

editions
  in CONNECT command, *4-2*

T

tablespace set, *5-3*
tablespaces
  containing XMLTypes, *1-3*
transporting data
  across platforms, *1-1*
  character sets, *1-3*
  compatibility considerations, *1-5*
  limitations, *1-3*
  national character sets, *1-3*
  transportable tablespaces
    limitations, *1-6*
    tablespace set, *5-3*
    transportable set, *5-2*
  XMLTypes in, *1-3*

X

XMLTypes
  transporting data, *1-3*