ORACLE

Full Transportable Export and Import

How to use Oracle Data Pump Full Transportable feature to migrate your Oracle Database

August, 2021 | Version 2.00

PURPOSE STATEMENT

This document provides an overview of Oracle Data Pump Full Transportable export and import.

DISCLAIMER

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

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

TABLE OF CONTENTS

INTRODUCTION

Full Transportable export and import can be used to migrate from earlier versions of Oracle Database to non-PDB and multi-tenent databases, making Oracle Database migrations faster, easier, and more efficient than ever before. Full transportable export is available with Oracle Database 11g Release 2 (11.2.0.3) and higher. Full transportable import is available with Oracle Database 12c (12.1.0.1) and higher.

This technical brief describes the full transportable export and import feature in Oracle Data Pump. After giving an overview of the benefits of using full transportable export/import, it explains how the feature works and provides a detailed example of full transportable export/import to show the syntax and process flow of using this feature. Note that, while this technical brief focuses on the use of full transportable export/import in a pluggable database environment, this feature can be used for migrations to a non-PDB database as well.

BENEFITS OF USING FULL TRANSPORTABLE EXPORT/IMPORT

Full transportable export/import combines the ease of use familiar to users of Data Pump export/import, with the speed of physical migration possible with transportable tablespaces. A recap of these migration techniques, and a comparison and contrast with full transportable export/import, helps show why full transportable export/import can be faster, simpler, and more efficient.

Oracle Data Pump export and import was introduced with Oracle Database 10g and is designed to handle large volumes of data. Oracle customers have easily exceeded data transfer rates of 10tb per hour. It employs techniques such as parallel worker processes, multiple dumpfiles and chooses the best access method for the data being moved. The command line for Oracle Data Pump offers a familiar feel to DBAs and is generally considered easy to use. As a logical migration method, Data Pump has maximum flexibility. It includes functionality for data and metadata compression, encryption, IOT to heap table conversion, object inclusion/exclusion, character set change, remapping tablespaces and schemas, transparent BasicFiles to SecureFiles conversion and more, and can even combine export and import into one operation over a DB link between source and target database. It easily moves a non-CDB database to multitenant.

There are cases where the physical migration method of transportable tablespaces can be faster than export/import with a dumpfile. For instance, when moving large amounts of data and/or moving large volumes of indexes - since they are not rebuilt - transportable tablespaces can be even faster.

Transportable tablespaces are usually the fastest fast way to move user and application data between databases. Tablespace data files are moved en masse from the source database to the target. Moving an entire data file is generally much faster than a logical export and import of individual rows or even blocks of data. However, traditional transportable tablespaces can require a fairly complicated set of steps to move the user and application metadata needed to use these tablespace data files in the destination database. A migration using transportable tablespaces can therefore be characterized as being very fast but more complex.

The best features of the logical and physical migration methods are combined in Full Transportable export/import, starting with Oracle Database 12c. The command line for full transportable export/import is Oracle Data Pump. Using Full transportable export/import you can take advantage of Data Pump options such as the ability to move metadata over a database link, and you are able to accomplish a full database migration with a single import command.

At the same time, full transportable export/import uses the transportable tablespaces mechanism to move user and application data. This results in a migration that is very fast, even for very large volumes of data. Most important, full

transportable export/import moves all of the system, user, and application metadata needed for a database migration, without the complex set of steps required for a traditional transportable tablespaces operation.

In summary, full transportable export/import combines the ease of use of Oracle Data Pump with the performance of transportable tablespaces, resulting in a feature that makes database migration faster and easier.

FULL TRANSPORTABLE EXPORT/IMPORT SUPPORT FOR PLUGGABLE DATABASES

Full transportable export/import supports Multitenant and pluggable databases (PDB). You can use it for bidirectional migrations: from a non-CDB database into a PDB and from a PDB to a non-CDB. It can also be used to migrate from one PDB to another PDB.

Overview of the process for migrating from a non-CDB into a pluggable database

1. Create a new PDB in the destination CDB using the `CREATE PLUGGABLE DATABASE` command
2. Set the user and application tablespaces in the source database to be `READ ONLY`
3. Copy the tablespace data files to the destination
4. Login to an account that has the `DATAPUMP_IMP_FULL_DATABASE` privilege and migrate using one of two methods:
   - Export/import from the source database using the `expdp` client with the `FULL=Y TRANSPORTABLE=ALWAYS` parameters in the command line, then import into the target PDB using the `impdp` client, or
   - Import over a database link from the source to the target using `impdp`
5. Perform post-migration database validation and application testing.

INTERNALS OF FULL TRANSPORTABLE EXPORT/IMPORT

This section provides an overview of the mechanisms used by full transportable export/import. This will help you gain a better understanding of what full transportable export/import does. Learing how full transportable operates helps answer the question about how it achieves the optimal combination of usability and performance.

Full Transportable Export/Import Concepts

To understand the internals of full transportable export/import, you need to know the difference between moving data in a *conventional* manner versus a *transportable* approach, and the distinction between administrative and user tablespaces.

When using *conventional* methods to move data, Oracle Data Pump uses either external tables or direct path unload to extract data. While the choice between these two access methods is based on the structure and types of the data being unloaded, both methods efficiently extract logical subsets of data from an Oracle database.

In contrast, a *transportable* move of data and indexes involves the physical movement of one or more tablespace data files. The data segments inside the tablespace data files are not read individually. Instead, the export operation extracts the metadata that describes the objects stored within each data file and moves each file as a single entity. Moving large volumes of data using transportable tablespaces can be faster than conventional data movement because there is no need to interpret and extract individual rows of data or index entries. It is possible to move individual tables or partitions in a transportable manner, but the entire tablespace data file is moved in these cases as well.

Understanding the difference between conventional and transportable data movement is helpful when considering the distinction between administrative and user tablespaces. For the purposes of a full transportable export, *administrative tablespaces* are the tablespaces provided by Oracle, such as `SYSTEM`, `SYSAUX`, `TEMP`, and `UNDO`. These tablespaces contain the procedures, packages, and seed data for the core Oracle Database functionality and Oracle-provided database components, such as Oracle Spatial, Oracle Text, OLAP, JAVAVM, and XML Database. In contrast, user tablespaces are those tablespaces defined by database users or applications. These may store user data, application data, and any other information defined by users of the database.

The first step for a full transportable export is creating the destination database. This newly created database includes a set of administrative tablespaces appropriate to the target environment, complete with Oracle-supplied components and packages. The user tablespaces should not exist for this destination databse. Tablespaces are created as part of the transportable import.

Starting with Oracle Database 12c, Oracle-supplied objects are neither exported nor imported by Oracle Data Pump. If there is user and/or application-defined data and metadata in a source database administrative tablespace, full transportable export extracts them using conventional data movement during the migration.

The user tablespaces, on the other hand, are moved to the destination database as full tablespace data files, under the assumption that a full export migrates all user and application data and metadata from the source to the destination system. User tablespaces are therefore moved in a transportable fashion, resulting in maximum performance for migrating user data.

Note: One consideration specific to full transportable export/import arises when a database object is stored across both user and administrative tablespaces, for instance a partitioned table. Storing an object in this way is generally not good practice, but it is possible. If there is an object with storage in both administrative and user tablespaces, then you can either redefine that object before transporting your data or use conventional Data Pump export/import. The example later in this technical brief shows how to detect this condition prior to starting a full transportable export.

Full Transportable Export

The key to the extra usability of full transportable export/import is extracting the metadata for all user and application-defined objects. A callout mechanism and API are provided for internal use by Oracle components, enabling migration of all metadata needed to create a full copy of the database during the import process.

All user and application objects that reside in administrative tablespaces are unloaded conventionally, both the metadata and data, for instance a table definition and the rows in that table. By contrast, objects stored in user tablespaces have only their metadata unloaded by Data Pump; the data for those objects is moved in a transportable fashion by copying the tablespace data files.

Invoking Full Transportable Export

Oracle Data Pump export client (expdp) is the command line interface for full transportable export. You can initiate a full transportable export by specifying two parameters in the parameter file or on the command line: `TRANSPORTABLE=ALWAYS` and `FULL=Y`. These parameter values tell Data Pump to use full transportable rather than conventional export methods.

Note: there is a special consideration if the `COMPATIBLE` database initialization parameter of the source database is not set to a value of at least 12.0. This would be true, for example, if you perform a full transportable export from Oracle Database 11g Release 2 (11.2.0.3). In this case, you must also specify the Data Pump parameter `VERSION=12.0` or higher. This denotes the fact that the result of the export will be imported into an Oracle Database 12c Release 1 (12.1) or later database.

Full transportable Import

Like a conventional Data Pump import, full transportable import can import a dump file or import directly from a source database into a destination database over a database link. The ability to import without using a dump file makes full transportable import an indispensable tool for database migrations.

Invoking Full Transportable Import

If you are importing a dump file, Oracle Data Pump is able to determine whether the dump file was produced by a conventional or full transportable export. A file-based full transportable import thus requires only that you specify the 1) dumpfile name and 2) the list of the copies of read-only user tablespace data files being transported using the `TRANSPORT_DATAFILES=<datafile_names>` parameter. This parameter can use a comma-separated list of files or can be specified multiple times for each data file.

Note: Oracle recommends using a parameter file to avoid syntax issues such as quotation marks around filenames.

If you are migrating directly from the source database to the target database without using a dump file, then you must specify an additional `NETWORK_LINK=<dblink_name>` parameter to start the operation in full transportable mode. This is because a network mode import actually performs the export from the source database and import to the target database as a single operation. A network-based full transportable import requires the following parameters:

- `FULL=Y, TRANSPORTABLE=ALWAYS` as specified for a full transportable export
- `TRANSPORT_DATAFILES=<datafile_names>` as used in a file-based full transportable import

- VERSION=12 or higher if the COMPATIBLE setting for the source database version is lower than 12.0. Note that the source database must be Oracle Database 11g Release 2 (11.2.0.3) or higher for a full transportable import
- NETWORK_LINK=<dblink_name> to specify the database link over which the data and metadata are transferred from the source database

Note: After successfully completing the import job all user tablespaces will be automatically set to a read/write state. If the original source data files are plugged into the destination database, they are no longer available to the source database. They belong to the destination database.

Endian Conversion

If both the source and target platforms have the same endian format, then data files can be transported as if they are on the same platform. If the source platform and the target platform have different endianness, then the data files must be converted to the target platform format using either the RMAN CONVERT command or the GET_FILE or PUT_FILE procedures in the DBMS_FILE_TRANSFER package.

Note: encrypted tablespaces cannot be transported to a platform with different endianness.

EXAMPLE: USING FULL TRANSPORTABLE TO MOVE A NON-CDB INTO A CDB

In this example, we use full transportable to migrate an Oracle Database 11g Release 2 (11.2.0.3) database from an Oracle Solaris x86 server to an Oracle Database 12c PDB in a CDB running on Oracle Linux. This example uses a dump file, requiring a full transportable export from the source database followed by a full transportable import at the destination database. The source database has two user tablespaces: hr_1 and hr_2. Note that tablespace HR_1 in this example is encrypted.

TABLE 1. SOURCE DATABASE TABLESPACES

TABLESPACE NAME	ENCRYPTED?	DATAFILE NAME
HR_1	Yes	/u01/app/oracle/oradata/hr_db/hr_101.dbf
HR_2	No	/u01/app/oracle/oradata/hr_db/hr_201.dbf

For this example, we assume that the Oracle SID of the target database is HR_PDB, and that the data files for the target PDB are stored in the directory /u01/app/oracle/oradata/hr_pdb/ on the destination server.

Note: The commands used to set up and administer Transparent Data Encryption (TDE) have changed after Oracle Database 11g Release 2. Please see Oracle Database Advanced Security Guide for more information about implementing TDE for your database.

STEP 1: Check the endianness of both platforms

To check the endianness of a platform, run the following query on each platform.

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

In this case, both Oracle Solaris x86 and Oracle Enterprise Linux have little endian format, so an endian conversion is not necessary.

STEP 2: Verify that the set of tablespaces to be transported is self-contained

Before transporting a set of tablespaces, you must verify that there are no logical or physical dependencies among the objects stored in the tablespaces being transported and those that are not being transported. For example, you need to detect situations in which an object is stored in both user and administrative tablespaces. This is referred to as a containment check.

To determine whether our tablespaces hr_1 and hr_2 are self-contained, including verification that referential integrity constraints will be valid after the transport, execute the following command on the source database.

```
SQL> EXECUTE DBMS_TTS.TRANSPORT_SET_CHECK('hr_1,hr_2', TRUE);
```

Note that you must include all user tablespaces in the database when performing this check for a full transportable export/import.

After invoking this PL/SQL procedure, you can see all violations by selecting from the TRANSPORT_SET_VIOLATIONS view.

```
SQL> SELECT * FROM TRANSPORT_SET_VIOLATIONS;
```

If the set of tablespaces is self-contained, this view will be empty. If any violations are listed, then you must address these issues before proceeding with the full transportable operation.

STEP 3: Create a directory object in the source database

Before you start the full transportable export, create a directory object into which the dump file will be written. Read and write access to this directory is automatically granted to the DBA role and to users SYS and SYSTEM.

```
SQL> CREATE DIRECTORY dp_dir AS '/u01/app/datafiles';
```

This directory object can be dropped from the database after the full transportable export has finished.

STEP 4: Place the hr_1 and hr_2 tablespaces in read-only mode

The tablespaces to be transported must be in read-only mode for the duration of the export. In this case we need to issue two commands on the source database.

```
SQL> ALTER TABLESPACE hr_1 READ ONLY;
SQL> ALTER TABLESPACE hr_2 READ ONLY;
```

The tablespaces can be returned to read-write status once the full transportable export has finished and the tablespace data files have been copied to the destination system.

STEP 5: Invoke full transportable export on the source database

Invoke the Data Pump export utility as a user with the DATAPUMP_EXP_FULL_DATABASE role.

```
$ expdp system/manager full=y transportable=always version=12 \
       directory=dp_dir dumpfile=full_tts.dmp \
       metrics=y exclude=statistics \
       encryption_password=secret123word456 \
       logfile=full_tts_export.log
```

Note: The use of METRICS=Y and EXCLUDE of statistics are are generally good practice for data pump exports. Beginning with Oracle Database 12g, LOGTIME=ALL is also recommended.

Notice that the VERSION=12 parameter is required because the source database is Oracle Database 11g Release 2 (11.2.0.3). This is the only time that a version number greater than the current version is allowed by the expdp command. If the source database is Oracle Database 12c or higher, with COMPATIBLE=12.0 or higher, then the VERSION parameter is not required.

After the export command completes, the export log file shows a list of all of the tablespace data files that need to be moved to the target.

STEP 6: Transport tablespace data files and the export dump file from source to target

Because we are migrating this database to a new server, we must copy the tablespace data files and the export dump file to a location that is accessible from the destination database. For example, on the destination system you could issue the following commands.

```
$ cd /u01/app/oracle/oradata/hr_pdb/

$ cp /net/<source-server>/u01/app/oracle/oradata/hr_db/hr_101.dbf .

$ cp /net/<source-server>/u01/app/oracle/oradata/hr_db/hr_201.dbf .

$ cp /net/<source-server>/u01/app/datafiles/full_tts.dmp  .
```

STEP 7: Create a directory object on the destination database

Because we copied the data pump dump file to the `oradata` directory for `HR_PDB`, we will create a directory object to point to that same directory for this import. This directory object must be created by a user connected to the PDB container.

```
SQL> CREATE DIRECTORY dp_dir AS '/u01/app/oracle/oradata/hr_pdb';

SQL> GRANT read, write on directory dp_dir to system;
```

This directory object can be dropped from the database after the full transportable import has finished.

STEP 8: Invoke full transportable import on the destination database

Invoke the Data Pump import utility as a user with `DATAPUMP_IMP_FULL_DATABASE` role.

```
$ impdp system/manager@hr_pdb directory=dp_dir \
        dumpfile=full_tts.dmp logfile=full_tts_imp.log \
        metrics=y \
        logtime=all \
        encryption_password=secret123word456 \
        transport_datafiles='/u01/app/oracle/oradata/hr_pdb/hr_101.dbf',\
                            '/u01/app/oracle/oradata/hr_pdb/hr_201.dbf'
```

Note: You must explicitly specify the service name of the PDB in the connect string for the impdp command.

Notice that, while this example shows several parameters specified on the command line, in most cases use of a data pump parameter file is recommended to avoid problems with blank space and quotation marks on the command line.

After this statement executes successfully, the user tablespaces are automatically placed in read-write mode on the destination database. Check the import log file to ensure that no unexpected error occurred and perform your normal post-migration validation and testing.

STEP 9: (Optional) Restore user tablespaces to read-write mode on the source database

After the full transportable export has finished, you can return the user-defined tablespaces to read-write mode at the source database if desired.

```
SQL> ALTER TABLESPACE hr_101 READ WRITE;

SQL> ALTER TABLESPACE hr_201 READ WRITE;
```

On the destination database, all tablespaces are set to read-write mode automatically upon completion of the full transportable import.

CONCLUSION

Full transportable export/import greatly simplifies and accelerates the process of database migration. Combining the ease of use of Oracle Data Pump with the performance of transportable tablespaces, full transportable export/import gives you the ability to upgrade or migrate to Oracle Database in a single operation if your source database is at least Oracle Database 11g Release 2 (11.2.0.3). Full transportable export/import is a valuable tool for migrating to pluggable databases, allowing you to take advantage of the cost savings and economies of scale inherent in moving to a multitenant architecture.

APPENDIX: LIMITATIONS ON FULL TRANSPORTABLE EXPORT/IMPORT

While full transportable export/import greatly enhances the usability of traditional transportable tablespaces, there are some limitations on transporting data about which you should be aware.

- Like traditional transportable tablespaces, full transportable export/import requires that the user tablespaces being migrated are in read-only mode for the duration of the export. If this is undesirable (which may be particularly true when you are testing this method before using it in production) you can use the RMAN capability to transport tablespaces from backups. See the *Oracle Database Backup and Recovery User's Guide* for more information on transporting tablespaces from backups.
- Objects with storage that are selected for export cannot straddle user and administrative ablespaces. They must have all of their storage segments entirely within user-defined transportable tablespaces or entirely within administrative, non-transportable tablespaces. Otherwise, transportable mode is inhibited and causes the export to fail. In this case, the export needs to be done conventionally or the object's storage redefined to meet this criterion.
- If the source and target platforms are of different endianness, then you must convert the data being transported so that it is in the format of the target platform.
- You can use the `RMAN CONVERT DATAFILE` command to convert the data.
- Alternatively data files can be converted to the target platform by transferring them with the `GET_FILE` or `PUT_FILE` procedures from the `DBMS_FILE_TRANSFER` package
- You cannot transport an encrypted tablespace to a platform with different endianness.
- To transport an encrypted tablespace to a platform with the same endianness, use the `expdp ENCRYPTION_PASSWORD` parameter during export. During import, specify the same value for the `impdp ENCRYPTION_PASSWORD` parameter.
- The XDB repository is not supported by full transportable export/import. However, user-defined XML schemas are supported by all forms of export/import: Data Pump export/import, transportable tablespaces, and full transportable export/import.
- The Automatic Workload Repository (AWR) is not supported by full transportable export/import. Use the `awrextr.sql` and `awrload.sql` scripts to move AWR data between Oracle databases.
- The source and target databases must use compatible database character sets. See the *Oracle Database Administrator's Guide* for more information about this and other General Limitations on Transporting Data.

CONNECT WITH US

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

 blogs.oracle.com

 facebook.com/oracle

 twitter.com/oracle

Oracle Database Full Transportable Export and Import
August, 2021
Author: Row Swonger
Contributing Authors: William Beauregard