Database Upgrade Using Transportable Tablespaces: Oracle Database 11*g* Release 1

Oracle Maximum Availability Architecture White Paper
February 2009

Maximum Availability Architecture

Oracle Best Practices For High Availability

ORACLE

Database Upgrade Using Transportable Tablespaces
Oracle Database 11*g* Release 1

Database Upgrade Using Transportable Tablespaces
Oracle Database 11*g* Release 1

INTRODUCTION

Efficient and reliable methods of performing database maintenance—such as upgrading a database—have existed for many Oracle software versions. However, as maintenance windows continue to shrink and database sizes continue to grow, the importance placed on the time required to upgrade a database to a new version has grown considerably.

The transportable tablespaces (TTS) feature, introduced in Oracle8*i*, was originally released as a method to move a subset of one database into another, such as plugging parts of an OLTP database into a data warehouse. However, TTS may reduce database upgrade time by moving all user tablespaces from a database running an earlier software release to an empty target database running a current software release. With the TTS feature, tablespace data files are plugged into the database by copying the data files to the target database, then importing the object metadata into the target database.

This white paper explains how to use the TTS feature in Oracle Database 11*g* Release 1 to upgrade a database from a previous Oracle Database release. This white paper complements the other MAA best practice papers that can be found on the MAA Web site on the Oracle Technology Network [1].

DATABASE UPGRADE USING TRANSPORTABLE TABLESPACES

This section describes database upgrade with transportable tablespaces under the following topics:

When to Use the TTS Method

If you cannot use Data Guard SQL Apply because of data type conflicts that Extended Datatype Support (EDS) cannot sufficiently resolve, and testing shows that upgrading with Database Upgrade Assistant (DBUA)[1] cannot meet uptime requirements, then consider upgrading a database using the TTS method that is described in this paper.

> **Note:** See the *Oracle Database High Availability Overview* [14] for a comprehensive description of the high availability solutions that prevent, tolerate, and reduce downtime for database upgrades and other types of planned maintenance.

The transportable tablespaces feature is an option for performing database upgrade in less than one hour for databases that have simple schemas and where the data files do not need to be transferred as part of the transport process (such as when the data files will be used in place).

This document only addresses upgrading a database. Regardless of the method you choose to perform the database upgrade, there are additional areas to consider to ensure a successful transition to the new software release. These areas include things such as understanding new features, changes in the new software, and developing a test plan. See the *Oracle Database Upgrade Guide* [2] for details.

Customer Examples

Numerous customers use TTS to successfully reduce database upgrade time:

- Amadeus, the leading Global Distribution System (GDS) and technology provider serving the marketing, sales and distribution needs of travel and tourism industries, reduced downtime when upgrading their Electronic Ticketing Server database from 25 minutes to 8 minutes. By using a physical standby database on new hardware, as described in the Best Practices section of this paper, Amadeus was able to eliminate the need to copy datafiles during the outage window, while maintaining the ability to fall back to the previous environment if a failure occurred during the upgrade. See the Amadeus High Availability case study [11] for more details.
- Another customer, who is one of the world's largest electrical engineering and electronics companies, accomplished a database upgrade in under three minutes. This customer also used a physical standby database on new hardware to eliminate datafile copying and to provide the ability to quickly fall back to the original environment if the TTS upgrade failed.

TTS has been used effectively to reduce database upgrade time. However, TTS was not originally designed as a database upgrade solution. Hence, TTS does not

[1] following the Oracle 11*g* Upgrade Companion available from My Oracle Support Note 601807.1

have the same level of automation and error checking found in Data Guard SQL Apply rolling upgrade or DBUA.

Guidelines for Using TTS

Consider the following guidelines when using TTS to upgrade a database:

- Upgrade with TTS requires a larger time investment to test the upgrade and to develop methods of validating the upgrade. Consider whether the added testing time and complexity of using a TTS upgrade are worth the potential to reduce downtime during the upgrade.
  - Best practices require thorough, full scope, and full size testing before attempting to use the upgraded database in production.
  - Application verification methods must be built, tested, and used to verify the TTS upgrade completed successfully.
- TTS requires a higher level of skill for both the database administrator and application administrator compared to DBUA and SQL Apply.
- TTS does not transport objects in the `SYSTEM` tablespace or objects owned by special Oracle users, like `SYS` or `SYSTEM`. Applications that store some of their objects in the `SYSTEM` tablespace or create objects as `SYS` or `SYSTEM` require additional steps and increase the complexity of the database upgrade.
- TTS, Data Pump, and traditional Export and Import do not import all system privileges into the upgraded database. An application that requires certain system privileges (for example, `SELECT` privilege on `SYS` view `DBA_USERS`) requires that all necessary privileges be granted in the target database. A script provided by the application vendor will simplify this step.
- TTS has documented restrictions that you must review. See the *Oracle Database Administrator's Guide* [3] for a list of TTS limitations.

Relationship of TTS to Data Pump

TTS works within the framework of Data Pump or the original Export/Import utilities. This white paper only discusses moving a database using TTS with Data Pump. If the source database is Oracle9*i* or earlier then you must use the original Export and Import utilities to transport the tablespaces from the source database to the target database. Although the steps performing the transport with Export and Import differ from those documented in this paper, the high-level steps presented remain accurate. See *Oracle Database Utilities* [4] for information about how to transport tablespaces using TTS with the original Export and Import.

Reducing DBUA Upgrade Time

DBUA performs an in-place upgrade of a database by running SQL scripts to upgrade both the database dictionary and database options installed within the database. Consider the following points to reduce upgrade time and possibly allow an upgrade with DBUA to fit within the desired maintenance window:

- Remove database options that are not being used. DBUA will upgrade all installed database options, whether or not they are required by an application. Reducing the number of options upgraded will reduce upgrade time.
- Remove unused user PL/SQL. All PL/SQL in the database is invalidated and recompiled as part of the upgrade process. Reducing the amount of recompilation required during the upgrade reduces upgrade time.
- Ensure the initialization parameters `CPU_COUNT` and `PARALLEL_THREADS_PER_CPU` have the correct value. PL/SQL recompilation occurs in parallel and uses the initialization parameter values of `CPU_COUNT` and `PARALLEL_THREADS_PER_CPU` to determine the degree of parallelism used.

For additional information about DBUA, see the *Oracle Database Upgrade Guide* [2].

Overview of the TTS Upgrade Process

Transporting a database using TTS involves performing the following steps:

Phase 1: Initial Setup

- Install the Oracle Database 11*g* release 11.1 software
- Run the pre-upgrade information tool
- Configure a physical standby database for the source database
- Handle objects in `SYSTEM` or `SYSAUX` tablespaces

Phase 2: Prepare the Source and Target Databases

- Gather information from the source database
- Create the target database
- Prepare the target and source databases
- Perform the self-containment check and resolve violations
- Export source database metadata

***Phase 3: Perform the Transport* (Downtime occurs during this step)**

- Prepare the source database for transport
- Stop Redo Apply and shut down the standby database
- Transport the user tablespaces

- Perform post-transport actions on the target database

Phase 4: Verify and Backup the New Target Database

- Verify the target database
- Backup the target database

See the *Oracle Database Administrator's Guide* [3] for more information about transporting tablespaces between databases, and *Oracle Database Utilities* [4] for more information about Data Pump, Export and Import.

Best Practices for the TTS Method

Upgrading a database by using transportable tablespaces works by creating a new database (called the target database) using the database software you are upgrading to, and then transporting all user tablespaces into the new target database. Initially, the target database consists only of the items necessary to permit the transport of all user data. When the transport operation occurs, copies of the datafiles from the source database are made available to the target database.

Use New Hardware for the Target Database

You must create and prepare the target database while the source database is still servicing users. While it is possible to create the target database on the same system as the source database, it is recommended that you use separate hardware to eliminate any impact on service to users while the environment is being setup for the TTS upgrade.

Use a Physical Standby to Eliminate Datafile Transfer Time

It is technically possible to use the source datafiles in place so that a large copy operation is unnecessary, but it is strongly discouraged because of the lack of a fallback unless a physical standby database is used for protection.

Create a physical standby database for the source database on the same system as the target database. The datafile location of the physical standby datafiles should be the final desired location for the target database datafiles. This eliminates the need to transfer or move datafiles during the TTS upgrade process.

- A physical standby database can be used to reduce the downtime required to perform a database upgrade by eliminating the need to transfer or copy the datafiles from the source location to the target location during the transport process. The standby datafiles are plugged directly into the target database during the transport process without first being copied. If a physical standby database is not used, datafiles from the source system will be copied and made available to the target system during the upgrade process, which will lengthen the time required for the upgrade.
- The source database can remain in tact during the upgrade and be used as a quick fallback should the upgrade fail.

Create a Fallback Plan

Because upgrading a database with TTS has a higher degree of risk compared to DBUA, you must place greater importance on having a well-prepared fallback plan. When using a physical standby database, as recommended, the standby datafiles are used in place. Once the transport operation begins, the source datafiles remain intact with all user tablespaces in `READ ONLY` mode. If the TTS upgrade fails, the tablespaces in the source database can be changed back to `READ WRITE` mode and users reconnected quickly so that downtime is minimized.

Use DBCA to Create the Target Database

The recommended method to create the target database is to use the Database Configuration Assistant (DBCA). DBCA templates can be used to create a new database from the configuration and structure of the existing source database.

Expected Upgrade Time

The initialization and preparation steps do not require downtime for the source database. The transport specific steps require downtime for the source database. The time required to perform the transport-specific steps varies significantly with the complexity and size of the database, the number of objects within the database, and the hardware and operating system. The following table describes the performance characteristics for major steps of the database transport phase of the process:

Step	Description
Export Tablespaces from Source Database	During the transportable export, the metadata is exported of the objects within the tablespaces being transported. The amount of time required to perform the transportable export depends on the number of objects being exported. To estimate transportable export time create a sample Data Pump parameter file with the `cr_tts_parfiles.sql` script provided in the Appendix, and then run the following Data Pump Export command: `$ expdp system/<password> parfile=dp_tsmeta_exp_TESTONLY.par` This command performs a nontransportable, metadata-only tablespace export of the tablespaces to be transported.
Make Source Tablespace Datafiles Available to Target Database	If following best practice recommendations, the amount of time required to make the source datafiles available to the target database should be minimal. During the transport process, the physical standby database datafiles are used in place; hence no datafile movement is necessary.
Import Tablespaces into Target Database	During the transportable import Data Pump imports the metadata of the objects within the tablespaces being transported. The amount of time required to perform the transportable import depends on the number of objects being imported.
Import Source Database Metadata into Target Database	Following the transportable import, the remaining metadata from the source database is imported into the target database, including user PL/SQL code. The length of time for this step depends on the number of objects being imported.
Compile invalid objects	Once the transport process is complete, invalid PL/SQL is recompiled.

Determine the Required Disk Space

The amount of additional disk space required for this process is equal to the sum of the following:

- The size of the source database for the physical standby database.
- The size of the `SYSTEM` and `SYSAUX` tablespaces and temporary tablespaces of the source system for the initial target database.
- One megabyte for each source database tablespace to serve as placeholders in the target database for the tablespaces being transported.

- The size of the export dump file containing the metadata of the tablespaces being transported from the source database to the target database.

PERFORMING A DATABASE UPGRADE USING TRANSPORTABLE TABLESPACES

Phase 1: Initial Setup

The initial setup phase involves installing the Oracle 11*g* Release 1 software on the target system and performing initial steps the source database to prepare for the transport process.

Patch set releases and critical patch updates can be obtained from My Oracle Support at http://myoraclesupport.oracle.com/.

Install the Oracle Database 11*g* release 11.1 Software

Install the Oracle Database 11*g* release 11.1 software, the latest patch set release, and the latest critical patch update on the target system.

Run the Pre-Upgrade Information Tool

Run the Pre-Upgrade Information Tool that is shipped with the Oracle Database 11.1 software on the source database. Connect with `SYSDBA` privileges to the source database running the earlier software release and run `utlu102i.sql` that is located in the target software version installation.

```
SQL> @<11.1_ORACLE_HOME>/rdbms/admin/utlu111i.sql
```

The Pre-Upgrade Information Tool identifies changes that must be made to the target database, which is running with later database software, based on the current settings of the source database, which is running an earlier release of software.

See the *Oracle Database Upgrade Guide* [2] for more information about the Pre-Upgrade Information Tool.

Configure a Physical Standby for the Source Database on the Target System

To reduce downtime by eliminating datafile copy time and to provide a fallback as protection from an upgrade failure, create a physical standby database for the source database on the same system as the target database. The datafile location of the physical standby datafiles should be the final desired location for the target database datafiles.

See *Oracle Data Guard Concepts and Administration* [5] for information about configuring a physical standby database.

Handle Objects in SYSTEM or SYSAUX

TTS does not move objects that reside in the `SYSTEM` or `SYSAUX` tablespaces of the source database.

There are three conditions that warrant special consideration when transporting a whole database:

- Metadata residing in the SYSTEM or SYSAUX tablespaces
- SYSTEM-owned objects residing in the SYSTEM or SYSAUX tablespaces
- User objects residing in the SYSTEM or SYSAUX tablespaces

Metadata Residing in the SYSTEM or SYSAUX Tablespaces

Database metadata includes views, synonyms, type definitions, database links, PL/SQL packages, roles, Java classes, privileges, sequences, and other objects. Running a full database, metadata-only import creates database metadata that is not automatically created in the target database by the transport process. This will be accomplished with two separate import processes, as detailed in the following steps.

SYSTEM-Owned Objects Residing in the SYSTEM or SYSAUX Tablespaces

Some applications create tables and indexes owned by the SYSTEM user that are required for proper application functionality. To properly identify these objects requires application-specific knowledge. You must move these objects to the target database manually with Data Pump or manually re-create the objects after performing the database upgrade.

User-Owned Tables Residing in the SYSTEM or SYSAUX Tablespaces

Run the following script (see the Appendix) to identify user objects that reside in the SYSTEM or SYSAUX tablespace:

```
SQL> @tts_system_user_obj.sql
```

You must move the identified objects to a user tablespace prior to beginning the transport process so the objects can be transported by TTS. Alternatively, you can move the objects separately with Data Pump or you can manually re-create them after performing the database upgrade.

Once this phase is started, no system privileges, tablespaces, users, or roles should be created, dropped, or modified in the source database.

Phase 2: Prepare the Source and Target Databases

This section describes tasks to complete preparations for upgrading a database using transportable tablespaces.

Gather Information from the Source Database

Certain information will be required from the source database that will be used throughout this process. The following scripts are provided in the Appendix:

Script	Description
cr_tts_drop_ts.sql	Creates `tts_drop_ts.sql` script from source database. The script is used to drop tablespaces in the target database prior to the transport process.
cr_tts_tsro.sql	Creates `tts_tsro.sql` script from the source database. The script is used to set all tablespaces to be transported to `READ ONLY` mode.
cr_tts_tsrw.sql	Creates `tts_tsrw.sql` script from the source database. The script is used to set all tablespaces to `READ WRITE` mode after the transport process.
cr_tts_sys_privs.sql	Creates `tts_sys_privs.sql` script from the source database. The script creates `GRANT` commands to be run on the target database to give privileges that are not handled by Data Pump.
cr_tts_create_seq.sql	Creates `tts_create_seq.sql` script from the source database. The script is used to reset the proper starting value for sequences on the target database.
cr_tts_parfiles.sql	Creates Data Pump parameter files for • XTTS export (`dp_ttsexp.par`) • XTTS import (`dp_ttsimp.par`) • Test tablespace metadata-only export (`dp_tsmeta_exp_TESTONLY.par`)

To gather the proper information from the source database, run the following cr_*.sql scripts:

```
SQL> connect system/<password>
SQL> @cr_tts_drop_ts.sql
SQL> @cr_tts_tsro.sql
SQL> @cr_tts_tsrw.sql
SQL> @cr_tts_sys_privs.sql
SQL> @cr_tts_parfiles.sql
```

Create the Target Database

Create a new database using the new software release. The new target database should be created on a different system to reduce impact on the source database during the preparation phase. The new target database consists initially of just `SYSTEM`, `SYSAUX`, `UNDO`, temporary tablespaces, and user tablespaces. The

recommended method of creating the target database is to use DBCA. When creating the target database, note the following:

- The Pre-Upgrade Information Tool, run in the previous step, identifies changes that must be accounted for with the new software version (for example, making initialization parameter changes).
- Although all user tablespaces from the original source database will be transported and plugged into the new target database during a later step in this white paper, initially the target database must contain the user tablespaces that are to 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 the same size or larger than those tablespaces on the source database.
- The sizes of log files and number of members per 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 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 `V$OPTION` to get currently installed database options.
    - Query `DBA_REGISTRY` to get currently installed database components.

Creating the Target Database with Database Configuration Assistant (DBCA)

Create the target database from the structure of the source database using the following four-step process:

1. Launch DBCA and click **Next** to continue to the Operations window.

   On the Operations window:

    a. Select **Manage Templates** and click **Next** to continue to the Template Management window.

    b. Select **From an existing database (structure only)** and follow the remaining windows to create a template of the existing source database. When complete, DBCA creates a template file as shown

in the following example, where *your_template_name* is the template name that you specified during template creation:

```
$ORACLE_HOME/assistants/dbca/templates/your_template_name.dbt
```

2. Reduce the size of the placeholder user tablespaces on the target database.

   When DBCA creates a database template, the tablespace names and data file sizes are the same as the source database. Although all user tablespaces from the original source database are transported and plugged into the new target database during a later step in this procedure, initially the target database must contain the user tablespaces that are to be transported. The size of the user tablespace data files initially created in the target database can be small and do not have to match the size in the source database. These placeholder tablespaces will be dropped prior to transporting in the tablespaces from the source system. By changing the size of the placeholder tablespaces, you can reduce the length of time it takes to create the target database.

   To create the target database with small placeholder user tablespaces, edit the template file that you created in step 1 (`$ORACLE_HOME/assistants/dbca/templates/mytemplate.dbt`). In each section labeled `DatafileAttributes`, edit the permanent tablespaces that contain user data to change the value of the `<size unit>` attribute to 1.

   > **Note:** Change *only* the `DatafileAttributes` sections for permanent tablespaces that contain user data. Do not change the `DatafileAttributes` section for any of the following tablespaces: `SYSTEM`, `SYSAUX`, any `UNDO` tablespaces, or any `TEMP` tablespaces.

For example, the following table compares the target database's original and updated template files for a single datafile:

Original Template File	Updated Template File
<DatafileAttributes id="/u01/app/oracle/oradata/tts11/users01.dbf"> <tablespace>USERS</tablespace> <temporary>false</temporary> <online>true</online> <status>0</status> **<size unit="MB">500</size>** <reuse>true</reuse> <autoExtend>true</autoExtend> <increment unit="KB">1280</increment> <maxSize unit="MB">32767</maxSize> </DatafileAttributes>	<DatafileAttributes id="/u01/app/oracle/oradata/tts11/users01.dbf"> <tablespace>USERS</tablespace> <temporary>false</temporary> <online>true</online> <status>0</status> **<size unit="MB">1</size>** <reuse>true</reuse> <autoExtend>true</autoExtend> <increment unit="KB">1280</increment> <maxSize unit="MB">32767</maxSize> </DatafileAttributes>

3. Move the template file to the following directory of the new ORACLE_HOME:

   $ORACLE_HOME/assistants/dbca/templates

4. With the environment set to the new ORACLE_HOME, launch DBCA and click **Next** to continue to the Operations window.

   On the Operations window:

   a. Select **Create a Database** to continue to the Database Templates window.

   b. On the Database Templates window, select the new template created from the structure of the source database.

   c. Continue through the remaining windows to create the target database based upon the structure of the existing source database.

There are other methods available to create the target database, such as modifying the CREATE DATABASE script that was originally used to create the source database.

See the *Oracle Database 2 Day DBA Guide* [6] for more information about using DBCA templates to create a new database.

Prepare the Target and Source Databases

Once the target database is created, it must be prepared for Data Pump usage and to accept the tablespaces being transported.

Create Database Link and Directory for Data Pump

On the target database, create a database link from the target system to the source system and a directory for Data Pump use.

```
SQL> connect system/<password>
SQL> create database link ttslink using 'staco07/orcl.us.oracle.com';
SQL> create directory ttsdir as '/u01/app/oracle/admin/orcl/tts';
SQL> !mkdir /u01/app/oracle/admin/orcl/tts
```

On the source database, create a directory for Data Pump use.

```
SQL> connect system/<password>
SQL> create directory ttsdir as '/u01/app/oracle/admin/orcl/tts';
SQL> !mkdir /u01/app/oracle/admin/orcl/tts
```

Create metadata required for TTS

The database metadata import is run in network mode and pulls directly from the source database.

Run Data Pump on the target system to import database metadata necessary for the transportable import.

```
$ impdp system/password DIRECTORY=ttsdir LOGFILE=dp_userimp.log
NETWORK_LINK=ttslink FULL=y INCLUDE=USER,ROLE,ROLE_GRANT,PROFILE
```

For additional information on Data Pump, see *Oracle Database Utilities* [4].

Drop user tablespaces

Drop the placeholder tablespaces in the target database that were created when the target database was initially created by DBCA. If the default permanent tablespace is one of the tablespaces that will be dropped from the target database because it will be transported, then first change the database default permanent tablespace.

```
SQL> select property_value
     from database_properties
     where property_name='DEFAULT_PERMANENT_TABLESPACE';

PROPERTY_VALUE
--------------
USERS

SQL> alter database default tablespace SYSTEM;
Database altered.
```

To drop all user tablespaces, run the `tts_drop_ts.sql` script created by running the `cr_tts_drop_ts.sql` script earlier.

```
SQL> @tts_drop_ts.sql
```

Perform the Self-Containment Check and Resolve Violations

The self-containment check, invoked by running the procedure `DBMS_TTS.TRANSPORT_SET_CHECK()`, ensures all object references from the transportable set are contained in the transportable set. For example, the base table of an index must be in the transportable set, index-organized tables and their overflow tables must both be in the transportable set, and a scoped table and its

base table must be together in the transportable set. Containment is usually less of an issue when transporting all user tablespaces. However, containment violations still can occur if there are references to user objects residing in the SYSTEM tablespace.

On the source database, run the tts_check.sql script (see the Appendix) to perform the self-containment check and report violations. Fix the violations reported before continuing.

```
SQL> @tts_check.sql
```

See Chapter 8 about "Managing Tablespaces" in the *Oracle Database Administrator's Guide* [3] for more information about the DBMS_TTS.TRANSPORT_SET_CHECK() PL/SQL procedure, running the self-containment check, and resolving containment violations.

NOTE: After performing this step, do not make any DDL changes to the source database. DDL changes made to the database after the source database metadata is exported will not be reflected in the target database unless handled manually.

Export Source Database Metadata

Export all metadata from the source database. After the tablespaces are transported, this metadata will be imported into target database to create metadata that was not transported. Do not perform any DDL changes after this step.

```
$ expdp system/password DIRECTORY=ttsdir LOGFILE=dp_fullexp_meta.log
DUMPFILE=dp_full.dmp FULL=y CONTENT=METADATA_ONLY
EXCLUDE=USER,ROLE,ROLE_GRANT,PROFILE
```

NOTE: The source database is unavailable to users from this point forward.

Phase 3: Perform the Transport

This section describes generating the steps required to complete the transport process.

Ready the Source Database for Transport

Perform the following steps on the source database to ready it for the transport process.

Application downtime begins with this step.

Shut Down the Application

Disconnect users and shutdown all application server processes. Users cannot use any application served by the database until the migration to the new platform is complete.

Export OLAP Analytic Workspaces

Follow the examples in My Oracle Support Note 352306.1 to perform OLAP AW export and import. While this document is primarily used for the migration of OLAP from 32-bit to 64-bit, the document properly describes how to migrate OLAP from one environment to another, regardless of bit size.

Oracle OLAP stores the OLAP `DECIMAL` data type in a hardware-dependent manner. Whenever changing platforms, all OLAP analytic workspaces (AWs) must be exported from the source database before the TDB process begins and imported into the target database after the TDB process is complete. AWs are exported and imported using the `DBMS_AW.EXECUTE` PL/SQL procedure. For additional information, see Oracle OLAP Reference [11] for Oracle Database 10*g*, or Oracle OLAP DML Reference for Oracle Database 11*g* [12].

Make All User Tablespaces READ ONLY

In the source database, place all user tablespaces in `READ ONLY` mode by running the `tts_tsro.sql` script (created when `cr_tts_tsro.sql` was run in phase 2).

```
SQL> @tts_tsro.sql
```

Gather Sequence Information

Proper sequence starting values need to be captured from the source database. This will be used to recreate sequences in the target database with the correct starting values. Run the script `cr_tts_create_seq.sql` (see the Appendix) on the source database to generate the SQL script `tts_create_seq.sql`, which contains `DROP SEQUENCE` and `CREATE SEQUENCE` statements to run on the target database in a later step.

```
SQL> @cr_tts_create_seq.sql
```

See the *Oracle Database Administrator's Guide* [3] for additional information about sequences.

Stop Redo Apply and Shut Down the Standby Database

The standby datafiles will be used directly. There is no need to perform a Data Guard switchover.

When using a physical standby database to facilitate the upgrade, issue the following statement to ensure that all archived redo log files have been applied to the standby database:

```
SQL> archive log list;
SQL> select sequence#, applied from v$archived_log order by sequence#;
```

Once all redo data has been received and applied to the standby database, stop Redo Apply, and shut down the standby instance.

```
SQL> alter database recover managed standby database cancel;
SQL> shutdown immediate;
```

Transport the User Tablespaces

Perform the following steps to perform the tablespace transport.

Export Tablespaces from the Source Database

Export the user tablespace metadata from the source database. The parameter file `dp_ttsexp.par` is created when `cr_tts_parfiles.sql` is run in phase 2.

If the datafiles must be copied to a new location or transferred to the target system because a physical standby database is not being used to facilitate this process, then the transportable export can proceed simultaneous with the datafile copy or transfer.

```
$ expdp system/password PARFILE=dp_ttsexp.par
```

This step may proceed simultaneous with the transportable export from the previous step.

Make Source Tablespace Datafiles Available to the Target Database

The recommended best practice is to create a physical standby database for the source database on the target system. Following this best practice reduces downtime and eliminates the need to copy or move datafiles. If a physical standby database has been created on the target system and the standby datafiles are properly located for use by the target database, then skip to the next step.

Do not use the source datafiles directly without having a fallback that can be utilized quickly in case of failure, such as a physical standby database.

Once the source tablespaces are placed in `READ ONLY` mode, the datafiles must be made available to the target database. If the target database resides on the same system as the source system then copy the datafiles to their new location.

If the target database resides on a different system from the source system, the datafiles being transported must be made available or moved from the source system to the target system. There are multiple ways to move the datafiles to the target system.

File system datafiles:

System commands, like ftp, that can transfer files using ASCII or binary mode, must transfer Oracle datafiles using binary mode.

- Use FTP or SCP to move the datafiles directly from the source system to the target system.
- NFS mount the filesystem containing the datafiles to the target system and copy the files to the target system.
- Reconfigure the SAN so that the storage devices can be mounted directly on the target system. Refer to your storage vendor for operating system specific details.

ASM datafiles:

- Reconfigure the SAN so that the storage devices can be mounted directly on the target system. Refer to your storage vendor for operating system specific details.
- Use the PL/SQL `DBMS_FILE_TRANSFER` package to transfer datafiles from source instance to target instance. See the *Oracle Database Administrator's Guide* [3] for details.
- Use XML DB FTP capability to ftp datafiles from source instance to target instance. See the *Oracle XML DB Developer's Guide* [7] for details.
- Use RMAN to move datafiles to a staging area on the source system, use standard operating system tools to transfer datafiles to the target system,

and then use RMAN to move the files into ASM on the target system. See the *Oracle Database Backup and Recovery User's Guide* [8] for details.

Copy Data Pump Dump Files to Target System

Copy to the target system the dump files created by Data Pump from the metadata export of the source database and the transportable export. The following example uses SCP to copy the dump files from the source system to the target system.

```
$ scp dp_full.dmp dp_tts.dmp target:/tmp
```

Import Tablespaces into Target Database

Import the user tablespaces into the target database.

Note: The `dp_ttsimp.par` file contains a list of datafiles that are to be transported into the target database. The contents of the file have been generated from the source database, including datafile names. You must change the datafile paths specified in the file to reflect the location where the datafiles exist on the target database. For example:

```
$ impdp system/password PARFILE=dp_ttsimp.par
```

Review the `tts_exp.log` file for errors.

Perform Post-Transport Actions on the Target Database

Make User Tablespaces READ WRITE on Target Database

After the transportable import, place user tablespaces in `READ WRITE` mode:

```
SQL> @tts_tsrw.sql
```

Import Source Database Metadata into Target Database

After the tablespaces are imported into the target database, import the remaining database metadata from the source database:

```
$ impdp system/password DIRECTORY=ttsdir LOGFILE=dp_fullimp.log
DUMPFILE=dp_full.dmp FULL=y
```

Review the `tts_dpnet_fullimp.log` file for errors. It is possible that errors or warnings can be ignored. However, you must investigate any reported errors and ignore errors only when you understand the source of the message and have assessed its impact.

There may be additional, application-specific privileges required. See your application vendor documentation to identify which scripts to run to create additional required privileges.

Create System Privileges in Target Database

Run `tts_sys_privs.sql` to create system privileges.

```
SQL> @tts_sys_privs.sql
```

Fix Sequence Values

Sequences may have values in the target database that do not match the source database because the sequences were referenced after the dictionary export was created. The supported method of resetting a sequence to a different starting value is to drop and recreate the sequence. Run the script `tts_create_seq.sql` (created in an earlier step in phase 3) to drop and re-create sequences based on the values in the source database. For example:

```
SQL> @tts_create_seq.sql
```

Compile Invalid Objects

Run `$ORACLE_HOME/rdbms/admin/utlrp.sql` to compile invalid objects.

```
SQL> @?/rdbms/admin/utlrp.sql
```

Import OLAP Analytic Workspaces

Use the examples in My Oracle Support Note 352306.1 to perform OLAP AW export and import. While this note is primarily used for the migration of OLAP from 32-bit to 64-bit, this document properly describes how to migrate OLAP from one environment to another regardless of bit size.

Import OLAP analytic workspaces (AWs) exported previously using the `DBMS_AW.EXECUTE` PL/SQL procedure. AWs are exported and imported using the `DBMS_AW.EXECUTE` PL/SQL procedure. For additional information, see Oracle OLAP Reference [12] for Oracle Database 10*g*, or Oracle OLAP DML Reference for Oracle Database 11*g* [13].

Phase 4: Verify and Backup the New Target Database

Once the transport process is finished, verify that the target database is complete and functional, and the target database is open and available. Once the target database and application verification completes successfully, users can connect for normal operation.

Gather Verification Information from Source Database

Following the transport process, validate the target database contents to ensure the necessary data and metadata exists for the application to run correctly. The complete information required for proper verification will differ depending on the application and database, but minimally should include a list of the following:

- Segment owners and types
- Object owners and types
- Invalid objects

There may be differences in the object counts between the source and target database that are acceptable provided they can be accounted for. For example, the target database may have fewer objects than the source database for a particular schema because of a table that cannot be transported with TTS.

The number and types of objects owned by SYS or SYSTEM or other Oracle internal schemas that are not transportable cannot be compared between two different database versions because of dictionary differences. These objects should not be used for verification between the source and target databases.

See the Appendix for the example script tts_verify.sql that can be run on the target database as an example of the information that may be required to compare the source and target databases. For example:

```
SQL> connect system/<password>
SQL> @tts_verify.sql
```

Perform Application-specific Verification

As indicated at the beginning of this document, it is necessary that you test application functionality against the target database prior to allowing full-scale use.

Verify and Gather Optimizer Statistics

Optimizer statistics may no longer be complete or accurate. Gathering new optimizer statistics may be necessary. See the *Oracle Database Performance Tuning Guide* [9] for details.

Backup the Target Database

Use Recovery Manager (RMAN) to backup the database so that physical and logical block validation is performed on all blocks in the transported datafiles.

Once verification has completed successfully, the final step is to perform a backup of the newly upgraded target database. Perform an online backup while the database is made available for use.

```
RMAN> backup check logical database;
```

Verify the Transported Datafiles

As an additional validation, run DBVERIFY against all transported datafiles to perform data and index block verification. DBVERIFY can have high I/O requirements, so database impact should be assessed before validating all transported datafiles, particularly if multiple DBVERIFY commands are run simultaneously.

```
$ dbv FILE=/oradata/ORCL/datafile/o1_mf_content_1wbq9rmd_.dbf
$ dbv FILE=/oradata/ORCL/datafile/o1_mf_users_1wbqls2r_.dbf
…
```

See *Oracle Database Utilities* [4] for additional information about DBVERIFY.

Start the Application

The final step is to start the application, directing connections to the database running on the new target platform.

CONCLUSION

Performing a database upgrade with transportable tablespaces may reduce downtime compared to using DBUA, but it does so with increased complexity and requires a greater testing effort to realize the potential gains.

APPENDIX

The appendix contains the scripts referenced in the steps to upgrade a database using transportable tablespaces.

cr_tts_sys_privs.sql is a sample script that creates `tts_sys_privs.sql` script **from the source database. Use this script to create GRANT commands to be run on the target database to give privileges that are not handled by Data Pump.**

cr_tts_sys_privs.sql

```
set heading off feedback off trimspool on escape off
set long 1000 linesize 1000
col USERDDL format A150
spool tts_sys_privs.sql
prompt /* ============ */
prompt /* Grant privs  */
prompt /* ============ */
select 'grant '||privilege||' on "'||
  owner||'".”'||table_name||'” to "'||grantee||’”’||
  decode(grantable,'YES',' with grant option ')||
  decode(hierarchy,'YES',' with hierarchy option ')||
  ';'
from dba_tab_privs
where owner in
      (select name
       from system.logstdby$skip_support
       where action=0)

  and grantee in
     (select username
      from dba_users
      where username not in
      (select name
       from system.logstdby$skip_support
       where action=0)     );
spool off
```

cr_tts_drop_ts.sqlis a sample script that creates `tts_drop_ts.sql` script **from source database. Use this script to drop tablespaces in the target database prior to the transport process.**

cr_tts_drop_ts.sql

```
set heading off feedback off trimspool on linesize 500
spool tts_drop_ts.sql
prompt /* ==================== */
prompt /* Drop user tablespaces */
prompt /* ==================== */
select ‘DROP TABLESPACE ‘ || tablespace_name ||
       ‘ INCLUDING CONTENTS AND DATAFILES;’
   from dba_tablespaces
   where tablespace_name not in ('SYSTEM','SYSAUX')
      and contents = ‘PERMANENT’;
spool off
```

cr_tts_tsro.sql is a sample script that creates `tts_tsro.sql` script **from the source database. Use this script to set all tablespaces to be transported to READ ONLY mode.**

cr_tts_tsro.sql

```
set heading off feedback off trimspool on linesize 500
spool tts_tsro.sql
prompt /* ================================= */
prompt /* Make all user tablespaces READ ONLY */
prompt /* ================================= */
select ‘ALTER TABLESPACE ‘ || tablespace_name || ‘ READ ONLY;’
   from dba_tablespaces
   where tablespace_name not in ('SYSTEM','SYSAUX')
      and contents = ‘PERMANENT’;
spool off
```

cr_tts_tsrw.sql is a sample script that creates `tts_tsrw.sql` script **from the source database. Use this script to set all tablespaces to READ WRITE mode after the transport process.**

cr_tts_tsrw.sql

```
set heading off feedback off trimspool on linesize 500
spool tts_tsrw.sql
prompt /* ================================== */
prompt /* Make all user tablespaces READ WRITE */
prompt /* ================================== */
select 'ALTER TABLESPACE ' || tablespace_name || ' READ WRITE;'
   from dba_tablespaces
   where tablespace_name not in ('SYSTEM','SYSAUX')
      and contents = 'PERMANENT';
spool off
```

cr_tts_create_seq.sql is a sample script that creates `tts_create_seq.sql` **script from the source database. Use this script to reset the proper starting value for sequences on the target database.**

cr_tts_create_seq.sql

```
set heading off feedback off trimspool on escape off
set long 1000 linesize 1000 pagesize 0
col SEQDDL format A300
spool tts_create_seq.sql
prompt /* ======================== */
prompt /* Drop and create sequences */
prompt /* ======================== */
select regexp_replace(
      dbms_metadata.get_ddl('SEQUENCE',sequence_name,sequence_owner),
      '^.*(CREATE SEQUENCE.*CYCLE).*$',
      'DROP SEQUENCE "'||sequence_owner||'"."'||sequence_name
                       ||'";'||chr(10)||'\1;') SEQDDL
   from dba_sequences
   where sequence_owner not in
      (select name
       from system.logstdby$skip_support
       where action=0)
;
spool off
```

cr_tts_parfiles.sql is a sample script that creates TTS export, TTS import, and test tablespace metadata-only export Data Pump parameter files.

cr_tts_parfiles.sql

```
REM
REM Create TTS Data Pump export and import PAR files
REM

set feedback off trimspool on
set serveroutput on size 1000000

REM
REM Data Pump parameter file for TTS export
REM
spool dp_ttsexp.par

declare
  tsname varchar(30);
  i number := 0;
begin
  dbms_output.put_line('directory=ttsdir');
  dbms_output.put_line('dumpfile=dp_tts.dmp');
  dbms_output.put_line('logfile=dp_ttsexp.log');
  dbms_output.put_line('transport_full_check=no');

  dbms_output.put('transport_tablespaces=');
  for ts in
    (select tablespace_name from dba_tablespaces
     where tablespace_name not in ('SYSTEM','SYSAUX')
       and contents = 'PERMANENT'
     order by tablespace_name)
```

```
  loop
    if (i!=0) then
      dbms_output.put_line(tsname||',');
    end if;
    i := 1;
    tsname := ts.tablespace_name;
  end loop;
  dbms_output.put_line(tsname);
  dbms_output.put_line('');
end;
/
spool off


REM
REM Data Pump parameter file for TTS import
REM
spool dp_ttsimp.par

declare
  fname varchar(513);
  i number := 0;
begin
  dbms_output.put_line('directory=ttsdir');
  dbms_output.put_line('dumpfile=dp_tts.dmp');
  dbms_output.put_line('logfile=dp_ttsimp.log');

  dbms_output.put('transport_datafiles=');
  for df in
    (select file_name from dba_tablespaces a, dba_data_files b
     where a.tablespace_name = b.tablespace_name
       and a.tablespace_name not in ('SYSTEM','SYSAUX')
       and contents = 'PERMANENT'
     order by a.tablespace_name)
  loop
    if (i!=0) then
      dbms_output.put_line(''''||fname||''',');
    end if;
    i := 1;
    fname := df.file_name;
  end loop;
  dbms_output.put_line(''''||fname||'''');
  dbms_output.put_line('');

end;
/
spool off


REM
REM Data Pump parameter file for tablespace metadata export
REM Only use this to estimate the TTS export time
REM
spool dp_tsmeta_exp_TESTONLY.par

declare
  tsname varchar(30);
  i number := 0;
begin
  dbms_output.put_line('directory=ttsdir');
  dbms_output.put_line('dumpfile=dp_tsmeta_TESTONLY.dmp');
  dbms_output.put_line('logfile=dp_tsmeta_exp_TESTONLY.log');
```

```
  dbms_output.put_line('content=metadata_only');

  dbms_output.put('tablespaces=');
  for ts in
    (select tablespace_name from dba_tablespaces
     where tablespace_name not in ('SYSTEM','SYSAUX')
       and contents = 'PERMANENT'
     order by tablespace_name)
  loop
    if (i!=0) then
      dbms_output.put_line(tsname||',');
    end if;
    i := 1;
    tsname := ts.tablespace_name;
  end loop;
  dbms_output.put_line(tsname);
  dbms_output.put_line('');
end;
/
spool off
```

tts_check.sql is a sample script that runs the DBMS_TTS.TRANSPORT_SET_CHECK function that performs the self containment check for the list of tablespaces to be transported.

tts_check.sql

```
declare
  checklist varchar2(4000);
  i number := 0;
begin
  for ts in
  (select tablespace_name
     from dba_tablespaces
     where tablespace_name not in ('SYSTEM','SYSAUX')
     and contents = 'PERMANENT')
  loop
    if (i=0) then
      checklist := ts.tablespace_name;
    else
      checklist := checklist||','||ts.tablespace_name;
    end if;
    i := 1;
  end loop;
  dbms_tts.transport_set_check(checklist,TRUE,TRUE);
end;
/
select * from transport_set_violations;
```

tts_system_user_obj.sqlis a sample script to identify user owned objects in the SYSTEM or SYSAUX tablespaces.

tts_system_user_obj.sql

```
select owner, segment_name, segment_type
  from dba_segments
  where tablespace_name in ('SYSTEM', 'SYSAUX')
    and owner not in
      (select name
       from system.logstdby$skip_support
       where action=0)
;
```

tts_verify.sql is a sample script to compare segment, object, and invalid object counts between the source and target databases.

tts_verify.sql

```
REM
REM Script to compare segment, object, and invalid object counts
REM between two databases.  This script should be run on the target
REM database.
REM
```

```
REM This script requires a database link named ttslink between the
REM source and target databases.
REM

set heading off feedback off trimspool on linesize 500

spool tts_verify.out

prompt
prompt Segment count comparison across dblink
prompt
select r.owner, r.segment_type, r.remote_cnt Source_Cnt, l.local_cnt
Target_Cnt
from ( select owner, segment_type, count(owner) remote_cnt
         from dba_segments@ttslink
        where owner not in
      (select name
       from system.logstdby$skip_support
       where action=0)        group by owner, segment_type ) r
    , ( select owner, segment_type, count(owner) local_cnt
          from dba_segments
         where owner not in
      (select name
       from system.logstdby$skip_support
       where action=0)         group by owner, segment_type ) l
where l.owner (+) = r.owner
  and l.segment_type (+) = r.segment_type
  and nvl(l.local_cnt,-1) != r.remote_cnt
order by 1, 3 desc
/

prompt
prompt Object count comparison across dblink
prompt
select r.owner, r.object_type, r.remote_cnt Source_Cnt, l.local_cnt
Target_Cnt
from ( select owner, object_type, count(owner) remote_cnt
         from dba_objects@ttslink
      (select name
       from system.logstdby$skip_support
       where action=0)        group by owner, object_type ) r
    , ( select owner, object_type, count(owner) local_cnt
          from dba_objects
         where owner not in
      (select name
       from system.logstdby$skip_support
       where action=0)         group by owner, object_type ) l
where l.owner (+) = r.owner
  and l.object_type (+) = r.object_type
  and nvl(l.local_cnt,-1) != r.remote_cnt
order by 1, 3 desc
/

prompt
prompt Invalid object count comparison across dblink
prompt
select l.owner, l.object_type, r.remote_cnt Source_Cnt, l.local_cnt
Target_Cnt
from ( select owner, object_type, count(owner) remote_cnt
         from dba_objects@ttslink
        where owner not in
      (select name
```

```
       from system.logstdby$skip_support
       where action=0)          and status='INVALID'
        group by owner, object_type ) r
    , ( select owner, object_type, count(owner) local_cnt
          from dba_objects
      (select name
       from system.logstdby$skip_support
       where action=0)          and status='INVALID'
         group by owner, object_type ) l
where l.owner = r.owner (+)
  and l.object_type = r.object_type (+)
  and l.local_cnt != nvl(r.remote_cnt,-1)
order by 1, 3 desc
/

spool off
```

REFERENCES

1. Oracle Maximum Availability Architecture (Web site)
   http://www.otn.oracle.com/goto/maa
2. *Oracle Database Upgrade Guide* (Part B28300)
   http://otn.oracle.com/pls/db111/db111.to_toc?partno=b28300
3. *Oracle Database Administrator's Guide* (Part B28310)
   http://otn.oracle.com/pls/db111/db111.to_toc?partno=b28310
4. *Oracle Database Utilities* (Part B28319)
   http://otn.oracle.com/pls/db111/db111.to_toc?partno=b28319
5. *Oracle Data Guard Concepts and Administration* (Part B28294)
   http://otn.oracle.com/pls/db111/db111.to_toc?partno=b28294
6. *Oracle Database 2 Day DBA* (Part B28301)
   http://otn.oracle.com/pls/db111/db111.to_toc?partno=b28301
7. *Oracle XML DB Developer's Guide* (Part B28369)
   http://otn.oracle.com/pls/db111/db111.to_toc?partno=b28369
8. *Oracle Database Backup and Recovery User's Guide* (Part B28270)
   http://otn.oracle.com/pls/db111/db111.to_toc?partno=b28270
9. *Oracle Database Performance Tuning Guide* (Part B28274)
   http://otn.oracle.com/pls/db111/db111.to_toc?partno=b28274
10. Database Rolling Upgrades Using Data Guard SQL Apply: Oracle Database 11*g* and 10*g*R2 (MAA white paper)
    http://www.oracle.com/technology/deploy/availability/pdf/maa_wp_10gr2_rollingupgradebestpractices.pdf
11. Amadeus High Availability Case Study
    http://www.oracle.com/technology/deploy/availability/pdf/AmadeusProfile_TTS.pdf
12. *Oracle OLAP Reference for Oracle Database 10g*
    http://otn.oracle.com/pls/db102/db102.to_toc?partno=b14350
13. *Oracle OLAP DML Reference for Oracle Database 11g*
    http://otn.oracle.com/pls/db111/db111.to_toc?partno=b28126
14. *Oracle Database High Availability Overview*
    http://otn.oracle.com/pls/db111/db111.to_toc?partno=b28281

Database Upgrade Using Transportable Tablespaces: Oracle Database 11*g* Release 1
February 2009
Author: Douglas Utzig
Contributors: Wei Hu, Alex Hwang, Alok Pareek, Viv Schupmann

Oracle USA, Inc.
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com