Platform Migration using Transportable Tablespaces: Oracle Database 10g Release 2

Oracle Maximum Availability Architecture White Paper
April 2007

Maximum Availability Architecture

Oracle Best Practices For High Availability

ORACLE

Platform Migration using Transportable Tablespaces
Oracle Database 10g Release 2

Database Upgrade using Transportable Tablespaces
Oracle Database 10g Release 2

INTRODUCTION

Efficient and reliable methods of performing database maintenance—such as migrating a database to a new platform—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 migrate a database to a more reliable or cost-effective platform has grown considerably.

Platform migration is the process of moving a database from one operating system platform to a different operating system platform. The supported way to accomplish this in prior releases of the Oracle database has been to export the data from the database on the old platform, create a new database on the new platform, and import the data into the new database. This process can take a number of days for a very large database. Oracle Database 10g provides two additional methods of migrating a database to a new platform. Oracle Database 10g Release 2 introduced Transportable Database (TDB), which is used to reduce the amount of time and effort required to migrate a database between platforms that share the same endian format (byte ordering). Oracle Database 10g Release 1 introduced cross platform transportable tablespaces (XTTS), which allows datafiles to be moved between platforms of different endian format.

XTTS is an enhancement to the transportable tablespace (TTS) feature introduced in Oracle8*i*. TTS 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 on the same platform. With the cross-platform enhancement, XTTS can plug datafiles into a database on a different platform, including those that are using a different endian format. XTTS may reduce platform migration time by moving all user tablespaces from a source database to an empty target database running on a platform that uses a different endian format. With the XTTS feature, tablespace datafiles plugged into the empty target database by copying the datafiles to the target database, converting them to the target system endian format, then importing the object metadata into the target database.

This white paper explains how to use the cross platform transportable tablespace feature in Oracle Database 10g Release 2 to migrate a database to a new platform that is using an endian format that is different from the source database platform.

This white paper complements other MAA best practice papers that can be found on the Oracle Technology Network [1].

PLATFORM MIGRATION USING TRANSPORTABLE TABLESPACES

When to Use This Method

If migrating to a platform that is the same endian format as the source platform (e.g. moving from a little endian platform to another little endian platform), the recommended migration method is Transportable Database (TDB). TDB significantly simplifies the platform migration process. For additional details, see the MAA best practice white paper Oracle Database 10g Release 2 Best Practices: Platform Migration using Transportable Database [2].

If migrating specifically between HP PA-RISC 64-bit and HP Itanium based servers running HP-UX v11, Oracle Database 10g Release 2 (and Oracle9i Database beginning with 9.2.0.7) supports a Data Guard configuration involving a primary database and one or more standby databases in a mixed HP architecture. Migrating between these platforms can be accomplished with a simple Data Guard switchover operation. See MetaLink Note 395982.1 for additional information.

The simplest and recommended method of migrating a database to a platform with a different endian format is to use Oracle Data Pump full database export and import. If the time it takes Data Pump to migrate the database to the new platform does not fit within the defined maintenance window even after following the Data Pump Performance guidelines in Oracle Database Utilities [3], consider using the XTTS method described in this paper.

This document is intended to address migrating a database only. Regardless of the method chosen to migrate to a new platform, there are additional areas that must be considered to ensure a successful transition to the new platform, such as understanding platform-specific features and changes in the Oracle software. Refer to the platform-specific installation guides, release notes, and READMEs [4] for details.

Guidelines for Using XTTS

If testing shows that migrating platforms with Data Pump cannot meet uptime requirements, migrating a database using XTTS can be considered when using the following guidelines:

- Migrating with XTTS requires a larger time investment to test the migration and to develop methods of validating the database and application. Consider whether the added testing time, complexity, and risk XTTS migration involves are worth the potential to reduce migration downtime.

- XTTS requires a higher level of skill for both the database administrator and application administrator compared to Data Pump full database export and import.
- XTTS 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 within the SYSTEM tablespace or create objects as SYS or SYSTEM require additional steps and increase complexity.
- XTTS, Data Pump, and traditional Export/Import will not import all system privileges into the upgraded database. An application that requires certain system privileges (e.g. SELECT privilege on SYS view DBA_USERS) will need all necessary privileges to be granted in the target database. A script provided by the application vendor will simplify this step.
- XTTS has documented restrictions that must be reviewed. Refer to the *Oracle Database Administrator's Guide* [5] for a list of XTTS limitations.

Relationship with Data Pump and Recovery Manager

XTTS works within the framework of Data Pump and Recovery Manager (RMAN). Data Pump is used to move to the target database the metadata of the objects within the tablespaces being transported. The original Export and Import utilities can also be used to transport tablespaces from one database to another, although the steps differ from those documented in this paper. Refer to *Oracle Database Utilities* [3] for information about how to transport tablespaces with Export and Import.

RMAN is used to convert the datafiles being transported to the endian format of the target platform. Refer to *Oracle Database Backup and Recovery Advanced User's Guide* [6] for information about using RMAN to convert datafiles to a different endian format.

Overview of the XTTS Upgrade Process

Transporting a database using XTTS involves performing the following steps:

Phase 1: Initial Setup

- Check Prerequisites
- Install the Oracle Database 10.2 Software
- Configure a Physical Standby for the Source Database (*if remote target database only*)
- Handle Objects in SYSTEM or SYSAUX

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 during this step)***

- Ready the Source Database for Transport
- Stop Redo Apply and Shut Down the Standby
- Transport the User Tablespaces
- Perform Post-Transport Actions on Target Database

Phase 4: Verify and Backup the New Target Database

- Verify Target Database
- Backup the Target Database

Refer to the *Oracle Database Administrator's Guide* [5] for more information about transporting tablespaces between databases, and to *Oracle Database Utilities* [3] for more information about Data Pump.

Best Practices

Migrating a database to a new platform using XTTS works by creating a new database (called the target database) on the target system, 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 converted to the new endian format and made available to the target database.

Optimize file conversion and transfer

The largest time and resource requirements are for manipulating the data files – copying them to the new system and converting them to the format needed for the new platform. RMAN does not convert datafiles to the new platform format in place; hence an essential requirement is the ability to store an extra complete copy of the data files on either the source or the target system. Decide which conversion location – source system or target system – will provide the least downtime.

XTTS requires that all datafiles be converted to the target platform endian format. The datafile conversion can occur on either the source system or on the target system. When performing a source system conversion, RMAN creates a converted copy of datafiles on the source system in the endian format of the target system.

The converted datafiles must then be transferred to, or otherwise be made available in, the proper location on the target system.

When performing XTTS with target system conversion, the original datafiles on the source system are first transferred to the target system and placed in a staging area. RMAN is then run to convert the datafiles to the target system endian format and place them in their final location.

When deciding the optimal way in which the datafiles will be made available to the target system, and on which system the conversion will be performed, consider the following:

- Datafile conversion is a highly I/O intensive operation that results in reading and writing the majority of the database, so most systems will find this to be the limiting resource. If one system has a significantly better I/O subsystem, perform the datafile conversion on that system.
- Use all available computer resources to do the datafile convert operations by running them in parallel. RMAN allows PARALLELISM to be specified with the CONVERT command so that multiple datafile conversions occur simultaneously. For additional information on RMAN parallelism and tuning RMAN, see the *Oracle Database Backup and Recovery Advanced User's Guide* [6].
- If datafiles will be transferred over the network from the source to the target system, datafile transfer and datafile conversion can occur simultaneously. For example, if performing target-system conversion, once a datafile is received completely on the target system, conversion can begin on that datafile while the next datafile is being received from the source system. The goal is to keep the network and disk capacity fully utilized simultaneously.
- Test the network link between the source and target systems with a bandwidth estimation tool (e.g. iperf, cap) to measure the bulk transfer capacity of the network link. High bandwidth and high latency links typically require changes to the default operating system parameters in order to achieve full utilization. Most importantly, set the send and receive socket buffer sizes to the bandwidth-delay product of the network link, and ensure that the tool used for the datafile transfer is using the larger value. Consult your operating system documentation for additional details of setting networking parameters for high bandwidth, high latency networks.
- Mount storage containing the datafiles on the target system. XTTS requires two operations on the datafiles: conversion to the target platform endian format, and transfer to the target system. Typically these operations occur as separate steps. For example, when performing a target system conversion, the datafiles are first transferred

to the target system, then converted using CONVERT in a separate step. However, these two operations can occur in a single step if the source datafile location is mounted on the target system.

- An example is rezoning a storage area network so that the volumes containing the datafiles are available to the target system (read only mode is sufficient). When the file conversion is run, the output of the RMAN CONVERT command places the datafiles in their final location without the need of a separate transfer step. The supportability of rezoning storage between platforms is operating system dependent. Contact your storage vendor for details.
- A second example is NFS mounting the source filesystem containing the datafiles on to the target system. When the file conversion is run on the target system, RMAN CONVERT reads from the NFS mount and places the converted datafiles in their final location without the need of a separate transfer step.
- The last example is NFS mounting the target filesystem that will be the final location of the converted datafiles on to the source system. When the conversion is run on the source system, RMAN CONVERT reads from the original datafile location and places the converted datafiles on the NFS mounted final location without the need of a separate transfer step.

Use a Physical Standby to Reduce Remote Datafile Transfer Time

If the target system is located remotely from the source system (i.e. it is not located on the same local area network), then create a physical standby database for the source database on a third system at the same location as the target system. The third system will be the same platform as the source system, as supported by Data Guard. Having a local copy of the datafiles at the same location as the target system eliminates the need to transfer datafiles across a high latency network during the XTTS process.

Create a Fall Back Plan

Because there is risk migrating to a new platform with XTTS, great importance must be placed on having a well-prepared fallback plan. During the XTTS process, tablespaces in the source database are placed in READ ONLY and copies of the datafiles are transferred to the target system. Since the source datafiles remain intact during the XTTS process, if the XTTS migration 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 leveraged to create a new database from the configuration and structure of the existing source database.

Expected Time Required

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, perform a non-transportable metadata only tablespace export of the tablespaces to be transported. A sample Data Pump parameter file can be created with the provided script cr_tts_parfiles.sql and run with the command: `$ expdp system/<password> parfile=dp_tsmeta_exp_TESTONLY.par`
Make source tablespace datafiles available to target database	Datafiles must be transferred to the target system. The time required is dependent on disk and network throughput capabilities of the source and target systems. If, following best practice recommendations, it is possible to utilize storage infrastructure to directly mount the source datafiles on the target system, then the time required for this step is minimal since no datafile movement is necessary.
Convert datafiles to target platform endian format	Datafiles must be converted to the new endian format. All datafiles being converted are read then written to a new location. Datafile conversion should fully utilize the available disk throughput.
Import tablespaces into target database	During the transportable import, the metadata is imported 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.

Step	Description
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.

Required Disk Space

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

- On the target system, disk space equal to the source database is required for the target database.
- If the datafiles will be transferred across the network then disk space equal to the size of the source database is required for the staging area. The staging area resides on the system where the conversion will take place.
    - If performing a source system conversion, CONVERT reads from the original datafile location and places converted datafiles in the staging area. The converted datafiles are transferred over the network to their final destination on the target system.
    - If performing a target system conversion, datafiles are transferred from their original location on the source system to the staging area on the target system. CONVERT reads from staging area and places converted datafiles in their final destination on the target system.
- If leveraging a physical standby database to assist in the migration to a remote location, disk space equal to the size of the source database is required.
- 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 export dump file containing the metadata of the tablespaces being transported from the source database to the target database.

Note: A staging area is not required if the original source datafile location can be mounted on the target system, or the final target datafile location can be mounted on source system.

PERFORMING A DATABASE UPGRADE USING TRANSPORTABLE TABLESPACES

Phase 1: Initial Setup

The initial setup phase involves installing the Oracle 10g Release 2 software on the target system and performing initial steps on the source database to prepare for the transport process.

Check Prerequisites

The following perquisites must be met prior to performing any cross-platform tablespace transport operation:

Determine if Source and Target Platforms are supported

Determine if XTTS is supported for both the source and target platforms, and determine the endian format (Little or Big) of each platform.

Run the following query to determine the platform name and endian format of the source database:

```
SQL> select d.platform_name, endian_format
     from v$transportable_platform tp, v$database d
     where tp.platform_name = d.platform_name;
```

Query `V$TRANSPORTABLE_PLATFORM` to determine support for the target platform:

```
SQL> select platform_name, endian_format
     from v$transportable_platform;
```

If the intended target platform is listed, then XTTS can be used for the platform migration. If the intended target platform is not listed, then the platform migration must be performed using a full database export and import using Data Pump.

If both platforms have the same endian format, TDB should be used for the platform migration. See the MAA best practice white paper Oracle Database 10g Release 2 Best Practices: Platform Migration using Transportable Database [2] for details.

Patch set releases and critical patch updates can be obtained from MetaLink at https://metalink.oracle.com.

Install the Oracle Database 10.2 Software

Install on the target system the same software version, patch set release, and critical patch update as the source system.

Configure a Physical Standby for the Source Database

If the target system is located remotely from the source system (i.e. it is not located on the same local area network), then create a physical standby database for the source database on a third system at the same location as the target system. The third system will be the same platform as the source system, as supported by Data Guard.

See *Oracle Data Guard Concepts and Administration* [7] for information about configuring a physical standby database. For Data Guard best practice recommendations, refer to the MAA best practice papers on the Oracle Technology Network [1].

Handle Objects in SYSTEM or SYSAUX

XTTS 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 will create 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 steps below.

SYSTEM-owned objects residing in the SYSTEM or SYSAUX tablespaces

Some applications create tables and indexes owned by SYSTEM that are required for proper application functionality. Proper identification of these objects must be done with application-specific knowledge. Moving these objects to the target database must be done manually with Data Pump or manual recreation following this procedure.

User-owned tables residing in the SYSTEM or SYSAUX tablespaces

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

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

The identified objects must be moved to a user tablespace prior to the transport process in order to be moved by XTTS. Alternatively, they can be moved separately with Data Pump or can be manually recreated following this procedure.

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 migrating a database to a new platform using XTTS.

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, which 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, which 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, which 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, which 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, which is used to reset the proper starting value for sequences on the target database.
cr_tts_parfiles.sql	Creates Data Pump parameters 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 on the target system. 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 DBCA. When creating the target database, note the following:

- 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 procedure, 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 and does 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.
- 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 have 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

Creating the target database from the structure of the source database is a two-step process. First, launch DBCA and click **Next** to continue to the Operations window. On the Operations window, select **Manage Templates** and click **Next** to continue to the Template Management window. Select **From an existing database (structure only)** and follow the remaining windows to create a template of the existing source database.

When DBCA creates a database template, the tablespace names and datafile sizes are the same as the source database. 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 procedure, initially the target database must contain the user tablespaces that are to be transported. The size of the user tablespace datafiles

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.

The recommended approach is to edit the `CreateDBFiles.sql` script created during template creation, and change the datafile size for all permanent tablespaces to 1M. For example,

Change lines like this:

```
CREATE SMALLFILE TABLESPACE "USERS" LOGGING DATAFILE SIZE 250M …
```

To this:

```
CREATE SMALLFILE TABLESPACE "USERS" LOGGING DATAFILE SIZE 1M …
```

After the source database template is created and the `CreateDBFiles.sql` script is modified as desired, launch DBCA and click **Next** to continue to the Operations window. On the Operations window, select **Create a Database** to continue to the Database Templates window. On the Database Templates window, select the new template created from the structure of the source database. Follow 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.

Refer to *Oracle Database 2 Day DBA Guide* [8] 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
```

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

Create metadata required for XTTS

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* [3].

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 when `cr_tts_drop_ts.sql` was run in phase 2.

```
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 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* [5] 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, no DDL changes are to be made to the source database. DDL changes made to the database after the source database metadata export 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 will be imported into target database to create metadata that was not transported. Perform no DDL 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

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

Disconnect Users and Restrict Access to Source Database

Restrict access to the source database and disconnect existing users to ensure there is no additional data modification.

```
SQL> alter system enable restricted session;
SQL> alter system disconnected session '<SID>,<SERIAL#>';
```

Refer to the *Oracle Database Administrator's Guide* [5] for additional information about placing the database in restricted mode and disconnecting current sessions.

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. The script `cr_tts_create_seq.sql` (see Appendix) can be run 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* [5] for additional information on sequences.

Stop Redo Apply and Shut Down the Standby

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

If using a physical standby database to facilitate with the upgrade, ensure that all archives have been applied to the standby database:

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

Once all archives have 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.

This step may proceed simultaneous with the datafile convert and transfer described in the next step.

Export Tablespaces from 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.

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

This step may proceed simultaneous with the transportable export described in the previous step.

Convert and Make Source Datafiles Available to Target Database

Once the source tablespaces are placed in `READ ONLY` mode, the datafiles must be made available to the target database. When moving a database to a platform that has a different endian format, the datafiles must be converted to the new format. The conversion can take place on either the source system before the datafiles are transferred to the target system, or on the target system after the datafiles are transferred to the target system.

If performing a source system conversion:

1. Run RMAN CONVERT TABLESPACE on the source system. Datafiles are read from their original source database location, and a converted copy of the datafile in the new endian format is placed in the staging area. All tablespaces being transported must be specified in the CONVERT command. The following example converts all datafiles in the specified tablespaces and places the converted datafile copies in the staging area /stage:

   When performing source system conversion, the `TO PLATFORM` clause is used to indicate the target system format.

   ```
   RMAN> CONVERT TABLESPACE DATA, IDX, USERS
     TO PLATFORM 'Linux IA (32-bit)'
     PARALLELISM 4
     DB_FILE_NAME_CONVERT '/oradata/ORCL/datafile/','/stage/';
   ```

   The sample script `cr_rman_ts_convert.sql` has been provided in the Appendix that can be run on the source database to generate an example, source-system conversion RMAN script. Review and edit the script for your environment, then run the script on the source system to perform the source system conversion.

2. Transfer the converted datafiles to their final destination on the target system. See "Moving the Datafiles" for additional information.

If performing a target system conversion:

1. Transfer the original datafiles to a staging area on the target system. See "Moving the Datafiles" for additional information.

2. Run RMAN CONVERT DATAFILE on the target system to convert the datafiles to the new endian format and place the converted copy in the final destination on the target system. The datafiles of all tablespaces being transported must be specified. The example below shows a target system conversion where the source datafiles have already been transferred to the target system into a staging area /stage. RMAN converts each datafile and places the converted datafile copies in their final target database destination within ASM:

When performing target system conversion, the `FROM PLATFORM` clause is used to indicate the source system format.

```
RMAN> CONVERT DATAFILE
        '/stage/o1_mf_data1_2k5c82h1_.dbf',
        '/stage/o1_mf_data2_2k5c8xmc_.dbf',
        '/stage/o1_mf_idx1_2k5c8zll_.dbf',
        '/stage/o1_mf_idx2_2jm6wwvt_.dbf',
        '/stage/o1_mf_users_2jm6trkj_.dbf'
  FROM PLATFORM 'Microsoft Windows IA (32-bit)'
  PARALLELISM 4
  DB_FILE_NAME_CONVERT '/stage/','+DATA';
```

The sample script cr_rman_df_convert.sql has been provided in the Appendix that can be run on the source database to generate an example, target-system conversion RMAN script. Review and edit the script for your environment, then run the script on the target system to perform the target system conversion.

Moving the Datafiles

There are multiple ways to accomplish moving 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.

- FTP or SCP 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 DBMS_FILE_TRANSFER package to transfer datafiles from source instance to target instance. Refer to the *Oracle Database Administrator's Guide* [5] for details.
- Use XML DB FTP capability to ftp datafiles from source instance to target instance. Refer to the *Oracle XML DB Developer's Guide* [9] 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. Refer to the *Oracle Database Backup and Recovery Advanced User's Guide* [6] 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. The datafile paths specified in the file must be changed to reflect the location where the datafiles exist on the target database.

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

Review the `tts_exp.log` file for errors.

Perform Post-Transport Actions on 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, the remaining database metadata from the source database can now be imported.

```
$ 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, errors reported must be investigated and ignored only when the source of the message is understood and its impact assessed.

There may be additional, application-specific privileges required. Please refer to your application vendor documentation to identify scripts to be 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. The script `tts_create_seq.sql`, created in an earlier step in phase 3, can be used to drop and recreate sequences based on the values in the source database.

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

Compile Invalid Objects

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

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

Phase 4: Verify and Backup the New Target Database

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

Gather Verification Information from Source Database

Following the transport process, the target database contents must be validated 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 should not be used for verification between the source and target databases.

See the Appendix for 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.

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

Perform Application-specific Verification

As indicated in the beginning of this document, it is necessary that application functionality be fully tested 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. Refer to the *Oracle Database Performance Tuning Guide* [10] 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. An online backup can be performed 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* [3] for additional information on DBVERIFY.

CONCLUSION

Performing a platform migration with XTTS may reduce downtime compared to a full database export and import, but does so with increased complexity requiring a greater testing effort to realize the potential gains.

APPENDIX

The appendix contains the scripts referenced in the steps to migrate a database using XTTS.

cr_tts_sys_privs.sql is a sample script that creates `tts_sys_privs.sql` script **from the source database, which creates 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
      ('SYS', 'SYSTEM', 'DBSNMP', 'SYSMAN', 'OUTLN', 'MDSYS',
       'ORDSYS', 'EXFSYS', 'DMSYS', 'WMSYS', 'WKSYS', 'CTXSYS',
       'ANONYMOUS', 'XDB', 'WKPROXY', 'ORDPLUGINS',  'DIP',
       'SI_INFORMTN_SCHEMA', 'OLAPSYS', 'MDDATA', 'WK_TEST',
       'MGMT_VIEW', 'TSMSYS')
  and grantee in
     (select username
      from dba_users
      where username not in
        ('SYS', 'SYSTEM', 'DBSNMP', 'SYSMAN', 'OUTLN', 'MDSYS',
         'ORDSYS', 'EXFSYS', 'DMSYS', 'WMSYS', 'WKSYS', 'CTXSYS',
         'ANONYMOUS', 'XDB', 'WKPROXY', 'ORDPLUGINS',  'DIP',
         'SI_INFORMTN_SCHEMA', 'OLAPSYS', 'MDDATA', 'WK_TEST',
         'MGMT_VIEW', 'TSMSYS')
     );
spool off
```

cr_tts_drop_ts.sqlis a sample script that creates `tts_drop_ts.sql` script **from source database, which is used 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, which is used 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, which is used 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, which is used 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
        ('SYS', 'SYSTEM', 'DBSNMP', 'SYSMAN', 'OUTLN', 'MDSYS',
         'ORDSYS', 'EXFSYS', 'DMSYS', 'WMSYS', 'WKSYS', 'CTXSYS',
         'ANONYMOUS', 'XDB', 'WKPROXY', 'ORDPLUGINS',  'DIP',
         'SI_INFORMTN_SCHEMA', 'OLAPSYS', 'MDDATA', 'WK_TEST',
         'MGMT_VIEW', 'TSMSYS');
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
```

cr_rman_ts_convert.sql is a sample SQL script that generates an RMAN script to perform source system conversion during XTTS platform migration.

cr_rman_ts_convert.sql

```
REM
REM Create RMAN CONVERT TABLESPACE script for cross platform TTS
REM Use for source system conversion only
REM

set feedback off trimspool on
set serveroutput on size 1000000

spool ts_convert.rman

declare
  tsname varchar(30);
  i number := 0;
begin
  dbms_output.put_line('# Sample RMAN script to perform file conversion
on all user tablespaces');
  dbms_output.put_line('# Tablespace names taken from
DBA_TABLESPACES');
  dbms_output.put_line('# Please review and edit before using');
  dbms_output.put_line('CONVERT TABLESPACE ');

  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('TO PLATFORM ''<Enter target platform here>''');
  dbms_output.put_line('PARALLELISM 4');
  dbms_output.put_line('DB_FILE_NAME_CONVERT
''/oradata/ORCL/datafile/'',''/stage/''');
  dbms_output.put_line(';');
end;
/
spool off
```

cr_rman_df_convert.sql is a sample SQL script that generates an RMAN script to perform target system conversion during XTTS platform migration.

cr_rman_df_convert.sql

```
REM
REM Create RMAN CONVERT DATAFILE script for cross platform TTS
REM Use for target system conversion only
REM

set feedback off trimspool on
set serveroutput on size 1000000

spool df_convert.rman

declare
  fname varchar(513);
  i number := 0;
begin
  dbms_output.put_line('# Sample RMAN script to perform file conversion
on all user datafiles');
  dbms_output.put_line('# Datafile names taken from DBA_DATA_FILES');
  dbms_output.put_line('# Please review and edit before using');
  dbms_output.put_line('CONVERT DATAFILE ');

  for df in
    (select substr(file_name,instr(file_name,'/',-1)+1) 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(''',');
    end if;
    i := 1;
    fname := df.file_name;
  end loop;
  dbms_output.put_line(''''||fname||'''');

  dbms_output.put_line('FROM PLATFORM ''<Enter source platform
here>''');
  dbms_output.put_line('PARALLELISM 4');
  dbms_output.put_line('DB_FILE_NAME_CONVERT ''/stage/'',''+DATA/''');
  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
        ('SYS', 'SYSTEM', 'DBSNMP', 'SYSMAN', 'OUTLN', 'MDSYS',
         'ORDSYS', 'EXFSYS', 'DMSYS', 'WMSYS', 'WKSYS', 'CTXSYS',
         'ANONYMOUS', 'XDB', 'WKPROXY', 'ORDPLUGINS',  'DIP',
         'SI_INFORMTN_SCHEMA', 'OLAPSYS', 'MDDATA', 'WK_TEST',
         'MGMT_VIEW', 'TSMSYS');
```

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
              ('SYS', 'SYSTEM', 'DBSNMP', 'SYSMAN', 'OUTLN', 'MDSYS',
              'ORDSYS', 'EXFSYS', 'DMSYS', 'WMSYS', 'WKSYS', 'CTXSYS',
              'ANONYMOUS', 'XDB', 'WKPROXY', 'ORDPLUGINS',  'DIP',
              'SI_INFORMTN_SCHEMA', 'OLAPSYS', 'MDDATA', 'WK_TEST',
              'MGMT_VIEW', 'TSMSYS')
```

```
        group by owner, segment_type ) r
    , ( select owner, segment_type, count(owner) local_cnt
          from dba_segments
         where owner not in
               ('SYS', 'SYSTEM', 'DBSNMP', 'SYSMAN', 'OUTLN', 'MDSYS',
               'ORDSYS', 'EXFSYS', 'DMSYS', 'WMSYS', 'WKSYS', 'CTXSYS',
               'ANONYMOUS', 'XDB', 'WKPROXY', 'ORDPLUGINS',  'DIP',
               'SI_INFORMTN_SCHEMA', 'OLAPSYS', 'MDDATA', 'WK_TEST',
               'MGMT_VIEW', 'TSMSYS')
         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
        where owner not in
              ('SYS', 'SYSTEM', 'DBSNMP', 'SYSMAN', 'OUTLN', 'MDSYS',
              'ORDSYS', 'EXFSYS', 'DMSYS', 'WMSYS', 'WKSYS', 'CTXSYS',
              'ANONYMOUS', 'XDB', 'WKPROXY', 'ORDPLUGINS',  'DIP',
              'SI_INFORMTN_SCHEMA', 'OLAPSYS', 'MDDATA', 'WK_TEST',
              'MGMT_VIEW', 'TSMSYS')
        group by owner, object_type ) r
    , ( select owner, object_type, count(owner) local_cnt
          from dba_objects
         where owner not in
               ('SYS', 'SYSTEM', 'DBSNMP', 'SYSMAN', 'OUTLN', 'MDSYS',
               'ORDSYS', 'EXFSYS', 'DMSYS', 'WMSYS', 'WKSYS', 'CTXSYS',
               'ANONYMOUS', 'XDB', 'WKPROXY', 'ORDPLUGINS',  'DIP',
               'SI_INFORMTN_SCHEMA', 'OLAPSYS', 'MDDATA', 'WK_TEST',
               'MGMT_VIEW', 'TSMSYS')
         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
              ('SYS', 'SYSTEM', 'DBSNMP', 'SYSMAN', 'OUTLN', 'MDSYS',
              'ORDSYS', 'EXFSYS', 'DMSYS', 'WMSYS', 'WKSYS', 'CTXSYS',
              'ANONYMOUS', 'XDB', 'WKPROXY', 'ORDPLUGINS',  'DIP',
              'SI_INFORMTN_SCHEMA', 'OLAPSYS', 'MDDATA', 'WK_TEST',
              'MGMT_VIEW', 'TSMSYS')
          and status='INVALID'
        group by owner, object_type ) r
    , ( select owner, object_type, count(owner) local_cnt
          from dba_objects
```

```
         where owner not in
               ('SYS', 'SYSTEM', 'DBSNMP', 'SYSMAN', 'OUTLN', 'MDSYS',
               'ORDSYS', 'EXFSYS', 'DMSYS', 'WMSYS', 'WKSYS', 'CTXSYS',
               'ANONYMOUS', 'XDB', 'WKPROXY', 'ORDPLUGINS',  'DIP',
               'SI_INFORMTN_SCHEMA', 'OLAPSYS', 'MDDATA', 'WK_TEST',
               'MGMT_VIEW', 'TSMSYS')
          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
   http://www.oracle.com/technology/deploy/availability/htdocs/maa.htm
2. Oracle Database 10g Release 2 Best Practices: Platform Migration using Transportable Database
   http://www.oracle.com/technology/deploy/availability/pdf/MAA_WP_10gR2_PlatformMigrationTDB.pdf
3. *Oracle Database Utilities* (Part B14215)
   http://otn.oracle.com/pls/db102/db102.to_toc?partno=b14215
4. Oracle Database Documentation Library, 10g Release 2 (10.2)
   http://www.oracle.com/pls/db102/portal.portal_db?selected=11
5. *Oracle Database Administrator's Guide* (Part B14231)
   http://otn.oracle.com/pls/db102/db102.to_toc?partno=b14231
6. *Oracle Database Backup and Recovery Advanced User's Guide* (Part B14191)
   http://otn.oracle.com/pls/db102/db102.to_toc?partno=b14191
7. *Oracle Data Guard Concepts and Administration* (Part B14239)
   http://otn.oracle.com/pls/db102/db102.to_toc?partno=b14239
8. *Oracle Database 2 Day DBA* (Part B14196)
   http://otn.oracle.com/pls/db102/db102.to_toc?partno=b14196
9. *Oracle XML DB Developer's Guide* (Part B14259)
   http://download-west.oracle.com/docs/cd/B19306_01/appdev.102/b14259/xdb22pro.htm#sthref2260
10. *Oracle Database Performance Tuning Guide* (Part B14211)
    http://otn.oracle.com/pls/db102/db102.to_toc?partno=b14211

Platform Migration using Transportable Tablespaces: Oracle Database 10g Release 2
April 2007
Author: Douglas Utzig
Contributing Authors: Wei Hu, Alex Hwang, Alok Pareek

Oracle Corporation
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