Migrate to Exadata Step by Step

Author:	ohsdba
Version:	V1.0
Creation Date:	October 2017

Confidential, Internal use only

Document Control

Major Contributors / Reviewers

Contributor	Role	Date	Contribution

Revision History

Issue Status	Author	Description	Date

目录

Introduction

There is no migration utility (Script or DBUA) to perform a cross platform migration of an Oracle Database. Changing platforms requires the database be re-built and / or the data moved using one of the following methods:

1. Export / Import to include the use of Datapump facilities. All versions support Export/Import but for Datapump 10.1.0.2 or higher is required
2. Transportable Tablespaces 10G or Later
3. RMAN Convert Database functions. 10G or Later
4. RMAN Duplicate
5. Streams Replication
6. Create Table As Select (CTAS)
7. Dataguard Heterogeneous Primary and Physical Standbys
8. Data Guard Transient Logical Standby
9. Oracle Golden Gate (For assistance with Oracle Golden Gate, an SR needs opened with the correct team)

Each available choice will have strengths and limitations to include data types, time required and potential costs. It will depend on BOTH the Operating System and Oracle versions on both the source and destination.

Generally speaking, Oracle XTTS is typically used when you are migrating from a platform with a different database endian (byte ordering) format and using Data Pump export/import does not meet the availability service level requirement.

Oracle Exadata Database Machine uses the Linux x86 64-bit operating system which is little endian format. XTTS can also be used to move from an older release of the database to a newer release starting with database release 10.2.0.3 on the source system.

When using Cross Platform Transportable Tablespaces (XTTS) to migrate data between systems that have different endian formats, the amount of downtime required can be substantial because it is directly proportional to the size of the data set being moved. However, combining XTTS with Cross Platform Incremental Backup can significantly reduce the amount of downtime required to move data between platforms.

To reduce the amount of downtime required for XTTS, Oracle has enhanced RMAN's ability to roll forward datafile copies using incremental backups, to work in a cross-platform scenario. By using a series of incremental backups, each smaller than the last, the data at the destination system can be brought almost current with the source system, before any downtime is required. The downtime required for datafile transfer and convert when combining XTTS with Cross Platform Incremental Backup is now proportional to the rate of data block changes in the source system.

The Cross Platform Incremental Backup feature does not affect the amount of time it takes to perform other actions for XTTS, such as metadata export and import. Hence, databases that have very large amounts of metadata (DDL) will see limited benefit from Cross Platform Incremental Backup since migration time is typically dominated by metadata operations, not datafile transfer and conversion.

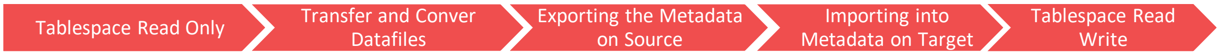
XTTS improvement

TTS feature available since Oracle 8i.Cross platform support since Oracle 10g.The Cross Platform Incremental Backup core functionality (incremental backup conversion) is delivered in Oracle Database 11.2.0.4 and later. If the target database is 11.2.0.4 or later, the target database can perform this function. If the destination database version is 11.2.0.3 or earlier, to perform incremental backup conversion, a separate 11.2.0.4 software home (the incremental convert home) must be installed, and an instance (the incremental convert instance), must be started in nomount state using that home. The incremental convert home and incremental convert instance are temporary and are used only during the migration.

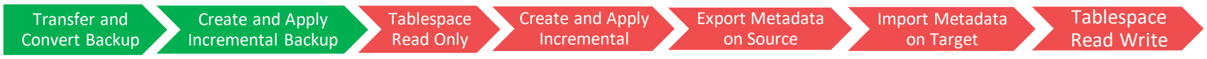
Version	Cross Platform	Different Block	Incremental Backups	Full transport
Oracle 8i				
Oracle 9i		√		
Oracle 10gR1	√	√		
Oracle 10gR2	√	√		
Oracle 11gR1	√	√		
Oracle 11gR2	√	√	√	√
Oracle 12c	√	√	√	√

🟩 Up Time 🟥 Down Time

Traditional XTTS


Tablespace Read Only
Transfer and Conver Datafiles
Exporting the Metadata on Source
Importing into Metadata on Target
Tablespace Read Write

Enhanced XTTS (target/stage database must be 11.2.0.4 afterwards)


Transfer and Convert Backup
Create and Apply Incremental Backup
Tablespace Read Only
Create and Apply Incremental
Export Metadata on Source
Import Metadata on Target
Tablespace Read Write

Traditional XTTS steps

- Put the tablespaces on source in read only mode(downtime begins)
- Transfer the datafiles to target
- Convert the datafiles as the target Endian format
- Export the metadata on source

- Import the metadata on target
- Put the tablespaces on target in read write mode

The source datafiles must be in read only before copying to the target system. This copy can take a very, very long time for a large, multi-terabyte database. That's problem if we do not have enough downtime.

Enhanced XTTS steps

- Transfer the datafiles from source to target(database is online)
- Convert the datafiles as target endian format
- Create incremental backup of the source tablespaces and transfer to target
- Convert the incremental backups as target endian format
- Apply the incremental backup to the target database
- Repeat the above steps in green color
- Put the tablespaces on source in read only mode(downtime begins)
- Do the final Steps in green color
- Export the metadata on source
- Import the metadata on target
- Put the tablespaces on target in read write mode

The initial copy of the datafiles occurs while the source database remains online. Then we took the incremental of the source database, transferred to the target, converted as target endian format, and applied to the target.

Migration method

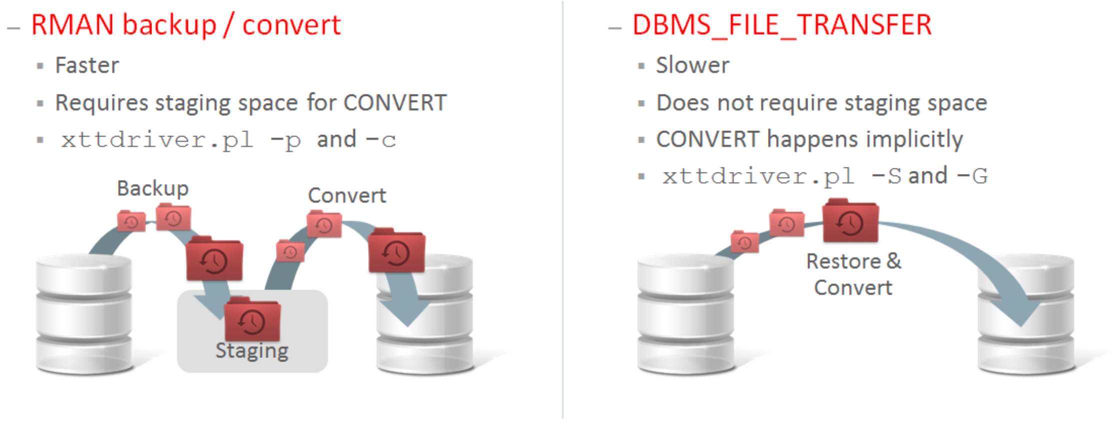
There are two primary scripts

- Perl script xttdriver.pl - the script that is run to perform the main steps of the XTTS with Cross Platform Incremental Backup procedure.
- Parameter file xtt.properties - the file that contains your site-specific configuration.

During the Prepare phase, datafiles of the tablespaces to be transported are transferred to the destination system and converted by the xttdriver.pl script. There are two possible methods:

1. Using dbms_file_transfer (DFT) transfer (using xttdriver.pl -S and -G options)
   It will generate two files
   xttnewdatafiles.txt
   getfile.sql
2. Using Recovery Manager (RMAN) RMAN backup (using xttdriver.pl -p and -c options)
   It will generate two files

xttplan.txt
rmanconvert.cmd



- RMAN backup / convert
  - Faster
  - Requires staging space for CONVERT
  - `xttdriver.pl -p` and `-c`

- DBMS_FILE_TRANSFER
  - Slower
  - Does not require staging space
  - CONVERT happens implicitly
  - `xttdriver.pl -S` and `-G`

Note: The rest phases are identical.

DBMS_FILE_TRANSFER

The dbms_file_transfer method uses the dbms_file_transfer.get_file() subprogram to transfer the datafiles from the source system to the target system over a database link. The dbms_file_transfer method has the following advantages over the RMAN method:

1) It does not require staging area space on either the source or destination system;

2) datafile conversion occurs automatically during transfer - there is not a separate conversion step. The dbms_file_transfer method requires the following:

- A destination database running 11.2.0.4. Note that an incremental convert home or instance do not participate in dbms_file_transfer file transfers.
- A database directory object in the source database from where the datafiles are copied.
- A database directory object in the destination database to where the datafiles are placed.
- A database link in the destination database referencing the source database.

RMAN

The RMAN backup method runs RMAN on the source system to create backups on the source system of the datafiles to be transported. The backups files must then be manually transferred over the network to the destination system. On the destination system the datafiles are converted by RMAN, if necessary. The output of the RMAN conversion places the datafiles in their final location where they will be used by the destination database. In the original version of xttdriver.pl, this was the only method supported. The RMAN backup method requires the following:

- Staging areas are required on both the source and destination systems for the datafile copies created by RMAN. The staging areas are referenced in the xtt.properties file using the parameters dfcopydir and stageondest. The final destination where converted datafiles are placed is referenced in the xtt.properties file using the parameter storageondest. Refer to the Description of Parameters in Configuration File xtt.properties section for details and sizing guidelines.

Details of using each of these methods are provided in the instructions below. The recommended method is the dbms_file_transfer method.

Parameters

	DFT	RMAN
tablespaces	√	√
platformid	√	√
srcdir	√	
dstdir	√	
srclink	√ (mandatory)	√ (optional)
dfcopydir		√
backupformat (incremental backup location, the parallel is depenad on rman device type disk parallelism)	√	√
stageondest		√
storageondest (final datafile location)	√	√
backupondest(incremental)	√	√
cnvinst_home(optional) (will be used if target database lower 11.2.0.4)	√	√
cnvinst_sid(optional) (will be used if target database lower 11.2.0.4)	√	√
asm_home(optional) (will be used if target database lower 11.2.0.4)	√	√
asm_sid(optional) (if backupondest is ASM DiskGroup, it will be used)	√	√
parallel (prepare phase, default 8)		√
rollparallel (parallelism for the -r roll forward operation)	√	√
getfileparallel (prepare phase, currently max value is 8)	√	
metatransfer (if metatransfer=1,it will transfer the temporary files and the backups from source to destination. desthost, desttmpdir needs to be defined)	√	√
destuser	√	√
desthost	√	√

desttmpdir	√	√
allowstandby (if allow, allowstandby=1)	√	√

Troubleshooting

To enable debug mode, either run xttdriver.pl with the -d flag, or set environment variable XTTDEBUG=1 before running xttdriver.pl. Debug mode enables additional screen output and causes all RMAN executions to be performed with the debug command line option.

Pre-requisites

The following prerequisites must be met before starting this procedure:

- The limitations and considerations for transportable tablespaces must still be followed. They are defined in the following manuals:
    - Oracle Database Administrator's Guide
    - Oracle Database Utilities
- In addition to the limitations and considerations for transportable tablespaces, the following conditions must be met:
    - The current version does NOT support Windows.
    - The source database must have its COMPATIBLE parameter set to 10.2.0 or higher.
    - The source database's COMPATIBLE parameter must not be greater than the destination database's COMPATIBLE parameter.
    - The source database must be in ARCHIVELOG mode.
    - Although preferred destination system is Linux (either 64-bit Oracle Linux or a certified version of RedHat Linux), this procedure can be used with other unix based operating systems. However, any non-Linux operating system must be on 11.2.0.4.
    - The Oracle version of source must be lower or equal to destination.
    - RMAN's default device type should be configured to DISK
    - RMAN on the source system must not have DEVICE TYPE DISK configured with COMPRESSED. If so, procedure may return: ORA-19994: cross-platform backup of compressed backups different endianness.
    - The set of tablespaces being moved must all be online, and contain no offline data files. Tablespaces must be READ WRITE. Tablespaces that are READ ONLY may be moved with the normal XTTS method. There is no need to incorporate Cross Platform Incremental Backups to move tablespaces that are always READ ONLY.

- All steps in this procedure are run as the oracle user that is a member of the OSDBA group. OS authentication is used to connect to both the source and destination databases.
- If the Prepare Phase method selected is dbms_file_transfer, then the destination database must be 11.2.0.4. See the *Selecting the Prepare Phase Method* section for details.
- If the Prepare Phase method selected is RMAN backup, then staging areas are required on both the source and destination systems. See the *Selecting the Prepare Phase Method* section for details.
- It is not supported to execute this procedure against a standby or snapshot standby databases.
- If the destination database version is 11.2.0.3 or lower, then a separate database home containing 11.2.0.4 running an 11.2.0.4 instance on the destination system is required to perform the incremental backup conversion. See the *Destination Database 11.2.0.3 and Earlier Requires a Separate Incremental Convert Home and Instance* section for details. If using ASM for 11.2.0.4 Convert Home, then ASM needs to be on 11.2.0.4, else error ORA-15295 (e.g. ORA-15295: ASM instance software version 11.2.0.3.0 less than client version 11.2.0.4.0) is raised.
- Tablespaces to be transported should not exist on target; if yes rename the tablespaces at target.
- Make sure that the schema users required for the tablespace transport exist in the target database.
- It's better to create database link on target to connect to source
- Enable block change tracking in source database
- Download supporting scripts for Cross Platform Incremental Backups from Doc ID 1389592.1/ 2005729.1
- dfcopydir, backupformat, stageondest and backupondest directories should be created before using the xtts migration tool and make sure it's enough to hold all backups or datafiles

Note: Only those database objects that are physically located in the tablespaces that are being transported will be copied to the destination system. If you need for other objects to be transported, that are located in different tablespaces (such as, for example, pl/sql objects, sequences, etc., that are located in the SYSTEM tablespace), you can use data pump to copy those objects to the destination system.

Building staging area

If you do not have enough space for staging, you can create ACFS volume on Exadata DiskGroup or create DBFS, and mount it on exadata compute nodes, then setup NFS on compute nodes as NFS Server.

On Exadata

mkdir /xttstage

```
chown -R oracle.oinstall /xttstage
chmod -R 755 /xttstage
mount /dev/asm-volume /xttstage
echo "/xttstage   *(rw,sync,no_root_squash)" >> /etc/exports
exportfs -r
showmount -e
```

On AIX

```
# nfso -a | grep nfs_use_reserved_ports
    nfs_use_reserved_ports = 0
# nfso -o nfs_use_reserved_ports=1
Setting nfs_use_reserved_ports to 1

mkdir /xttstage
chown -R oracle:oinstall /xttstage
mount -o rw,bg,hard,noac,nointr,proto=tcp,vers=3,rsize=131072,wsize=131072,timeo=600
192.168.10.12:/xttstage /xttstage
```

Database check

Check archive log on source

```
sqlplus / as sysdba
archive log list;
```

Check whether need do recovery

```
select * from v$recover_file;
```

Check database character

Make sure both databases are using the same character set and national character set

```
select parameter,value
from nls_database_parameters
where parameter in
('NLS_CHARACTERSET','NLS_NCHAR_CHARACTERSET');
```

	source	target
CHARACTERSET		
National CHARACTERSET		

Check supported platform

Make sure both platform are supported platforms

```
select * from v$transportable_platform order by platform_id;
```

Check endian format

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

	source	target
Endian Format		

- if source db = BIG and TARGET = BIG _ No RMAN convert
- if source db = LITTLE and TARGET = LITTLE _ No RMAN convert
- if they different _ RMAN convert is required

Check platform id and name

Check what is the platforms on both databases, since this will be used later

```
select platform_id,platform_name from v$database;
```

	source	target
platform_id		
platform_name		

```
set lines 156
column PLATFORM_NAME format a30
select PLATFORM_NAME, ENDIAN_FORMAT FROM V$TRANSPORTABLE_PLATFORM WHERE
PLATFORM_ID = ( SELECT PLATFORM_ID FROM V$DATABASE );

PLATFORM_NAME                  ENDIAN_FORMAT
------------------------------ --------------
AIX-Based Systems (64-bit)     Big

SQL>

set lines 156
column PLATFORM_NAME format a30
select PLATFORM_NAME, ENDIAN_FORMAT FROM V$TRANSPORTABLE_PLATFORM WHERE
PLATFORM_ID = ( SELECT PLATFORM_ID FROM V$DATABASE );

PLATFORM_NAME                  ENDIAN_FORMAT
------------------------------ --------------
Linux x86 64-bit               Little
```

Check self-containment

Perform self-containment check on the required tabelsapce/s, to see what kind of violation you might encounter, and fix them if necessary.

```
execute dbms_tts.transport_set_check('t1,t2',true);
select * from transport_set_violations;
```

Check external tables, directories, or BFILE(11g+)

```
set serveroutput on;
declare x boolean;
begin
        x := dbms_tdb.check_external;
end;
/
```

Check Index Organized Tables

Index Organized Tables (IOT) can become corrupt when using Transportable Tablespace (TTS) from Solaris, Linux or AIX to HP/UX. Currently there is no patch for this issue, the Index Organized Tables (IOT) need to be recreated after the TTS.

References : Document 371556.1 How to Move Tablespaces Across Platforms Using Transportable Tablespaces With RMAN, Bug 9816640 closed as not feasible to fix.

Affected Version : All Versions currently (check enhancement Bug 12683199 for version implemented).

```
select owner,table_name,iot_type from dba_tables where iot_type like '%IOT%';
```

Check XML Type

Beginning with Oracle Database 10g Release 2, you can transport tablespaces that contain XMLTypes, but you must use the IMP and EXP utilities, not Data Pump; when using EXP, ensure that the CONSTRAINTS and TRIGGERS parameters are set to Y (the default). For versions 11.1 and higher, you must use only Data Pump, not the IMP and EXP Utilities. This restriction on XML types and tables with binary XML storage continues in 12.1.

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

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

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

Check spatial indexes

Restriction : Be aware that TTS across different endian platforms are not supported for spatial indexes in 10gR1 and 10gR2; such a limitation has been released in 11g
- specific Spatial packages must be run before exporting and after transportation, please see Oracle Spatial documentation.

Reference : Oracle Spatial Developer's Guide
Affected Version : All versions

Transportable Tablespace import using IMPDP fails when the tablespace contains a spatial index defined in it.

Document 579136.1 IMPDP TRANSPORTABLE TABLESPACE FAILS for SPATIAL INDEX
Document 2031174.1 TRANSPORTABLE TABLESPACE IMPORT FAILS WITH ORA-39083 ORA-942 ON MDRS TABLES

Check Materialized Views or Partitioned Tables

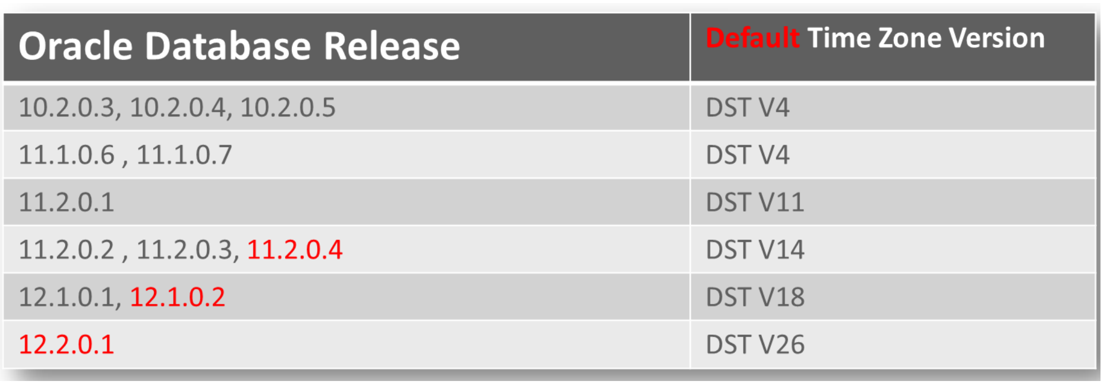
Objects with underlying objects (such as materialized views) or contained objects (such as partitioned tables) are not transportable unless all of the underlying or contained objects are in the tablespace set.

Check Timezone

If the source is Oracle Database 11g release 2 (11.2.0.2) or later and there are tables in the transportable set that use TIMESTAMP WITH TIMEZONE (TSTZ) columns, then the time zone file version on the target database must exactly match the time zone file version on the source database.

If the source is earlier than Oracle Database 11g release 2 (11.2.0.2), then the time zone file version must be the same on the source and target database for all transportable jobs regardless of whether the transportable set uses TSTZ columns.

If these requirements are not met, then the import job aborts before anything is imported. This is because if the import job were allowed to import the objects, there might be inconsistent results when tables with TSTZ columns were read.



Oracle Database Release	Default Time Zone Version
10.2.0.3, 10.2.0.4, 10.2.0.5	DST V4
11.1.0.6 , 11.1.0.7	DST V4
11.2.0.1	DST V11
11.2.0.2 , 11.2.0.3, 11.2.0.4	DST V14
12.1.0.1, 12.1.0.2	DST V18
12.2.0.1	DST V26

To identify the time zone file version of a database, you can execute the following SQL statement:

```
SQL> SELECT VERSION FROM V$TIMEZONE_FILE;
```

And

This select gives all TimeStamp with Time Zone (TSTZ) columns in your database:

```
select c.owner || '.' || c.table_name || '(' || c.column_name || ') -'
    || c.data_type || ' ' col
  from dba_tab_cols c, dba_objects o
 where c.data_type like '%WITH TIME ZONE'
    and c.owner=o.owner
   and c.table_name = o.object_name
   and o.object_type = 'TABLE'
   And o.owner not in ('SYS', 'SYSTEM')
order by col
/
```

If source and target timezone file is not same, you can recreate the target database as source version. Before recreating, export the ORA_TZFILE as source.

```
$ export ORA_TZFILE=$ORACLE_HOME/oracore/zoneinfo/timezlrg_13.dat
```

Check options

```
set linesize 156 pages 156
select parameter,value from v$option order by 2;

PARAMETER                                     VALUE
----------------------------------------------------------------- ------------
Real Application Clusters                        FALSE
Automatic Storage Management                          FALSE
Oracle Label Security                           FALSE
ASM Proxy Instance                              FALSE
Unified Auditing                               FALSE
Management Database                                 FALSE
I/O Server                                    FALSE
Oracle Database Vault                            FALSE
Partitioning                                  TRUE
```

Objects	TRUE
Advanced replication	TRUE
Bit-mapped indexes	TRUE
Connection multiplexing	TRUE
Connection pooling	TRUE
Database queuing	TRUE
Incremental backup and recovery	TRUE
Instead-of triggers	TRUE
Parallel backup and recovery	TRUE
Parallel execution	TRUE
Parallel load	TRUE
Point-in-time tablespace recovery	TRUE
Fine-grained access control	TRUE
Proxy authentication/authorization	TRUE
Change Data Capture	TRUE
Plan Stability	TRUE
Online Index Build	TRUE
Coalesce Index	TRUE
Managed Standby	TRUE
Materialized view rewrite	TRUE
Database resource manager	TRUE
Spatial	TRUE
Export transportable tablespaces	TRUE
Transparent Application Failover	TRUE
Fast-Start Fault Recovery	TRUE
Sample Scan	TRUE
Duplexed backups	TRUE
Java	TRUE
OLAP Window Functions	TRUE
Block Media Recovery	TRUE
Fine-grained Auditing	TRUE
Application Role	TRUE
Enterprise User Security	TRUE
Oracle Data Guard	TRUE
OLAP	TRUE
Basic Compression	TRUE
Join index	TRUE
Trial Recovery	TRUE
Advanced Analytics	TRUE
Online Redefinition	TRUE
Streams Capture	TRUE
File Mapping	TRUE
Block Change Tracking	TRUE
Flashback Table	TRUE
Flashback Database	TRUE
Transparent Data Encryption	TRUE
Backup Encryption	TRUE
Unused Block Compression	TRUE
Result Cache	TRUE
SQL Plan Management	TRUE
SecureFiles Encryption	TRUE

```
Real Application Testing                          TRUE
Flashback Data Archive                            TRUE
DICOM                                        TRUE
Active Data Guard                               TRUE
Server Flash Cache                              TRUE
Advanced Compression                                TRUE
XStream                                      TRUE
Deferred Segment Creation                            TRUE
Exadata Discovery                               TRUE
Data Mining                                   TRUE
Global Data Services                           TRUE
Adaptive Execution Plans                           TRUE
Table Clustering                               TRUE
Zone Maps                                      TRUE
Real Application Security                         TRUE
Privilege Analysis                             TRUE
Data Redaction                                  TRUE
Cross Transportable Backups                          TRUE
Cache Fusion Lock Accelerator                        TRUE
Snapshot time recovery                             TRUE
Heat Map                                      TRUE
Automatic Data Optimization                          TRUE
Transparent Sensitive Data Protection                  TRUE
In-Memory Column Store                               TRUE
Advanced Index Compression                            TRUE
In-Memory Aggregation                               TRUE

86 rows selected.

SQL>
```

Check additional

If the owner/s of tablespace objects does not exist on target database, the usernames need to be created manually before starting the transportable
tablespace import.

Advanced Queues Transportable tablespaces do not support 8.0-compatible advanced queues with multiple recipients.

Opaque Types Types(such as RAW, BFILE, and the AnyTypes) can be transported, but they are not converted as part of the cross-platform transport operation. Their actual structure is known only to the application, so the application must address any endianness issues after these types are moved to the new platform.

Floating-Point Numbers BINARY_FLOAT and BINARY_DOUBLE types are transportable using Data Pump but not the original export utility, EXP.

Before performing a TTS procedure ii, it's important to be aware that the use of traditional EXP/IMP with 11.2 is desupported. Original Export is desupported for general use as of Oracle Database 11g. The only supported use of original Export in Oracle Database 11g is backward migration of XMLType data to Oracle Database 10g release 2 (10.2) or earlier. Therefore, Oracle recommends that you use the new Data Pump Export and Import utilities, except in the following situations which require original Export and Import.

Encrypted tablespaces have the following limitations

Before transporting an encrypted tablespace, you must copy the Oracle wallet manually to the destination database, unless the master encryption key is stored in a Hardware Security Module (HSM) device instead of an Oracle wallet. When copying the wallet, the wallet password remains the same in the destination database. However, it is recommended that you change the password on the destination database so that each database has its own wallet password. See Oracle Database Advanced Security Administrator's Guide for information about HSM devices, about determining the location of the Oracle wallet, and about changing the wallet password with Oracle Wallet Manager.

You cannot transport an encrypted tablespace to a database that already has an Oracle wallet for transparent data encryption. In this case, you must use Oracle Data Pump to export the tablespace's schema objects and then import them to the destination database. You can optionally take advantage of Oracle Data Pump features that enable you to maintain encryption for the data while it is being exported and imported. See Oracle Database Utilities for more information.

You cannot transport an encrypted tablespace to a platform with different endianness.

Tablespaces that do not use block encryption but that contain tables with encrypted columns cannot be transported. You must use Oracle Data Pump to export and import the tablespace's schema objects. You can take advantage of Oracle Data Pump features that enable you to maintain encryption for the data while it is being exported and imported. See Oracle Database Utilities for more information.

Database Charactersets Compatibility between Source and Destination

The source and the destination databases must use compatible database character sets. That is, one of the following must be true:

- The database character sets of the source and the target databases are the same.
- The source database character set is a strict (binary) subset of the target database character set, and the following three conditions are true:
  + The source database is in version 10.1.0.3 or higher.
  + The tablespaces to be transported contain no table columns with character length semantics or the maximum character width is the same in both the source and target database character sets.
  + The tablespaces to be transported contain no columns with the CLOB data type, or the source and the target database character sets are both single-byte or both multibyte.
- The source database character set is a strict (binary) subset of the target database character set, and the following two conditions are true:
  + The source database is in a version lower than 10.1.0.3.
  + The maximum character width is the same in the source and target database character sets.

https://mikedietrichde.com/2016/05/25/transportable-tablespaces-characters-sets-same-same-but-different/

Prepare exadata

Cleanup all unnecessary files and database before go live, and create the new database, modify database initial parameters. Make sure it's in best state.

Pre-health check on source and target

Make sure source and target database in health status

Migration summary

We will migrate tablespaces from 10.2.0.5 Database running on IBM-AIX 5.3 Power with Big Endian format to 12c Database running on Oracle Exadata (OEL 6.9) with Small Endian format. We will use RMAN mode in this document.

Time summary

Migration Section	Time Markers	Done
Prerequisites, Known Issues and Limitations of XTTS	15 Minutes	Yes
Preparing the Source Database		
Preparing the Target Database		
Phase 1 Initial Setup phase		
Phase 2 Prepare phase		
Phase 3 Roll Forward phase		
Phase 4 Transport Phase: Export the source schema metadata		
Phase 4 Transport Phase: Importing schema metadata into target		
Phase 4 Transport Phase: Import datafiles		
Phase 5 Validate the Transported Tablespaces		
Post Health Check		

Source and target hardware info

	Source	Target

CPU	2 x RAC Node	2 Node RAC
Hardware	IBM P590	Exadata
Endian	BIG	LITTLE
Operating System	AIX 5.3	OEL 6.9
Disk Group	ASM	ASM
DB Size	6+ TB	6+ TB
Database Version	10.2.0.5	12.1.0.2

Source and target OS info

AIX	**Server1**	**Server2**
Running DB Instances		
Purpose (Production, development, Q&A)		
Platform		
Model		
Operating System		
O/S Version and Release		
# CPU		
Processor / CPU Speed		
Total Physical Memory		
Backup on disk/tape		
Backup Device name/model		
Backup Device Size		
Backup time		

Exadata	**Server1**	**Server2**
Running DB Instances		
Purpose (Production, development, Q&A)		

Platform		
Model		
Operating System		
O/S Version and Release		
# CPU		
Processor / CPU Speed		
Total Physical Memory		
Backup on disk/tape		
Backup Device name/model		
Backup Device Size		
Backup time		

Source and target database info

	AIX	Exadata
Hosts		xd06dbadm01 xd06dbadm02
DB Unique Name		
DB Version		
Container DB		
Instance Name		
SCAN Name		
Scan Port		
Diskgroup		
ORACLE_HOME		
ASM_HOME		
Endian format		
Platform id		
Platform name		
Character		
National Character		
Timezone (select dbtimezone from dual)		
Timezone File Version (select version from v$timezone_file)		
Blocksize		

Sessions		
Processes		
Components		
Inmemory_size		

```
col comp_id for a10
col version for a12
col comp_name for a35
select comp_id,comp_name,version,status from dba_registry;

COMP_ID    COMP_NAME                           VERSION      STATUS
---------- ----------------------------------- ------------ -------------------
CATALOG    Oracle Database Catalog Views       12.2.0.1.0   VALID
CATPROC    Oracle Database Packages and Types  12.2.0.1.0   VALID
JAVAVM     JServer JAVA Virtual Machine        12.2.0.1.0   VALID
XML        Oracle XDK                          12.2.0.1.0   VALID
CATJAVA    Oracle Database Java Packages       12.2.0.1.0   VALID
APS        OLAP Analytic Workspace             12.2.0.1.0   VALID
RAC        Oracle Real Application Clusters    12.2.0.1.0   OPTION OFF
XDB        Oracle XML Database                 12.2.0.1.0   VALID
OWM        Oracle Workspace Manager            12.2.0.1.0   VALID
CONTEXT    Oracle Text                         12.2.0.1.0   VALID
ORDIM      Oracle Multimedia                   12.2.0.1.0   VALID
SDO        Spatial                             12.2.0.1.0   VALID
XOQ        Oracle OLAP API                     12.2.0.1.0   VALID
OLS        Oracle Label Security               12.2.0.1.0   VALID
DV         Oracle Database Vault               12.2.0.1.0   VALID

15 rows selected.

SQL>
```

Component ID	**Component Name**	**Version**	**Status**

Transport Tablespace on source

select distinct tablespace_name from dba_segments where owner='<>';

Owner (s)	
Tablespace(s)	

Check objects on source

select count(*) from dba_objects where owner='';

select owner,object_type,object_name from dba_objects where owner='' and status='INVALID' order by owner,object_type;

select owner,object_type,count(*) from dba_objects where owner='' group by owner,object_type order by object_type;

	source
Owner Invalid objects	
Invalid objects summary	
Objects summary	

Check pluggable database on target

```
set linesize 156 pagesize 156
col name for a10
select name,dbid,con_uid,guid,open_mode from v$pdbs;

NAME            DBID    CON_UID GUID                             OPEN_MODE
---------- ---------- ---------- -------------------------------- ----------
PDB$SEED   3367002637 3367002637 760270D22244401B933A5144B8F0D553 READ ONLY
PDB3       4139058161 4139058161 3E188A0579E943B1B936A034712FA069 MOUNTED
PDB1       3015784063 3015784063 7C7273A42B0348539458967FACB0A41A MOUNTED
PDB2       4196478222 4196478222 E21399C38A4E4F0D9AA356BCF207B302 MOUNTED

SQL>
```

Phase 1 - Initial Setup

Prerequisites verification

Please make sure your environment satisfies all the above mentioned prerequisites.

Create directory on source and target

```
sqlplus / as sysdba
create directory dpump as '/xttstage '; (on source and target)
```

Create xtts directory on source

```
mkdir /xttstage/xtts
upload and unzip rman_xttconvert_v3.zip
cd /xttstage/xtts
unzip rman_xttconvert_v3.zip
Archive: rman_xttconvert_v3.zip
inflating: xtt.properties
inflating: xttcnvrtbkupdest.sql
inflating: xttdbopen.sql
inflating: xttdriver.pl
inflating: xttprep.tmpl
extracting: xttstartupnomount.sql
```

Note: As we put the xtts scripts on stage directory (it's NFS directory), just unzip it on source.

Modify environment

```
export TMPDIR=/xttstage/xtts
```

Note: add above line on source and target profile

Configure xtt.properties

```
tablespaces=TS1,TS2
platformid=6
srclink=
dfcopydir=/xttstage
backupformat=/xttstage
stageondest=/xttstage
storageondest=+DATA
backupondest=/xttstage
parallel=6
rollparallel=8
```

Note: storageondest is the final location where you put the datafiles. If pluggable database is used, you'd better put them under pdb directory.
storageondest=+DATA/pcdb/760270D22244401B933A5144B8F0D553

Enable block change tracking on source

```
alter database enable block change tracking using file '<name>';
```

Purge recyclebin

```
sqlplus / as sysdba
purge dba_recyclebin;
```

Create dblinks on target

```
create public database link ttslink connect to system identified by Welcome1 using
'(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL = TCP)(HOST = 192.168.10.11)(PORT
= 1521)))(CONNECT_DATA = (SERVICE_NAME = ptgdb)))';
```

Phase 2 – Prepare phase

Run prepare script on source

```
./xttdriver.pl -p
```

This script will generate two files
xttplan.txt (record the current database scn)
rmanconvert.cmd (convert command)

Output

Convert the datafiles on target

```
./xttdriver.pl -c
```

Output

Phase 3 - Roll Forward Phase

Create an incremental backup on source

```
./xttdriver.pl -i
```

It will generate two files
tsbkupmap.txt

incrbackups.txt (record the backupsets location)

Output

Convert, apply the incremental backup on target

./xttdriver.pl -r

Output

Determine the from scn for the next incremental backup

./xttdriver.pl -s

The determine new FROM_SCN step calculates the next FROM_SCN, records it in the file xttplan.txt, then uses that SCN when the next incremental backup is created

Output

Repeat roll forward steps in order if necessary

./xttdriver.pl -i

Output

./xttdriver.pl -r

Output

NOTE: If a datafile is added to one a tablespace since last incremental backup and/or a new tablespace name is added to the xtt.properties, the following will appear:

```
Error:
------
The incremental backup was not taken as a datafile has been added to the tablespace:
```

```
Please Do the following:
--------------------------
1. Copy fixnewdf.txt from source to destination temp dir

2. Copy backups:
<backup list>
from <source location> to the <stage_dest> in destination

3. On Destination, run $ORACLE_HOME/perl/bin/perl xttdriver.pl --fixnewdf

4. Re-execute the incremental backup in source:
$ORACLE_HOME/perl/bin/perl xttdriver.pl --bkpincr

NOTE: Before running incremental backup, delete FAILED in source temp dir or
run xttdriver.pl with -L option:

$ORACLE_HOME/perl/bin/perl xttdriver.pl -L --bkpincr

These instructions must be followed exactly as listed. The next incremental backup will include the
new datafile.
```

Phase 4 - Transport Phase

Put tablespaces in read only on source

alter tablespace TS1 read only;

Create the final incremental backup on source

./xttdriver.pl -i

Output

Convert and apply incremental backup on target

./xttdriver.pl -r

Output

Import owner metadata on target

Import through dmp file

expdp "'/ as sysdba'" directory=dpump schemas= content=metadata_only exclude=table,index,statistics dumpfile= logfile= parallel= filesize= metrics=yes job_name = migrate_meta

impdp "'/ as sysdba'" directory=dpump schemas= dumpfile= logfile= parallel= metrics=yes job_name = migrate_meta

Import through network_link

impdp "'/ as sysdba'" directory=dpump network_link= ttslink schemas= content=metadata_only exclude=table,index,ststictics logfile=impdp_metadata_golive.log

Import object metadata into target database

You can import the transport datafiles into pdb (pluggable database) /non-cdb database as before!!!!

impdp """/ as sysdba"""
expdp \'/ as sysdba\'

./xttdriver.pl -e
It will generate file xttplugin.txt

Output

You have three ways to do the import, just choose one (import traditional is desupported since 11gR2)

Import use network_link

- The TRANSPORT_TABLESPACES parameter is valid only when the NETWORK_LINK parameter is also specified.
- Transportable mode does not support encrypted columns.
- When using the IMPDP utility, and the option TRANSPORT_FULL_CHECK. This parameter is valid for transportable mode (or table mode when TRANSPORTABLE=ALWAYS was specified on the export) only when the NETWORK_LINK parameter is specified.

impdp directory=DATA_PUMP_DIR logfile=tts_imp.log network_link=ttslink \
transport_full_check=no \

```
exclude=table_statistics,index_statistics \
transport_tablespaces=TS1,TS2 \
transport_datafiles='+DATA/prod/datafile/ts1.285.771686721', \
'+DATA/prod/datafile/ts2.286.771686723', \
'+DATA/prod/datafile/ts2.287.771686743'
```

Output

Import manually

When using The EXPDP utility, and the option TRANSPORT_TABLESPACES

- Transportable jobs are not restartable.
- Transportable jobs are restricted to a degree of parallelism of 1.
- Transportable tablespace mode requires that you have the DATAPUMP_EXP_FULL_DATABASE role.
- Transportable mode does not support encrypted columns.
- The default tablespace of the user performing the export must not be set to one of the tablespaces being transported.
- The SYSTEM and SYSAUX tablespaces are not transportable.
- All tablespaces in the transportable set must be set to read-only.
- If the Data Pump Export VERSION parameter is specified along with the TRANSPORT_TABLESPACES parameter, then the version must be equal to or greater than the Oracle Database COMPATIBLE initialization parameter.
- The TRANSPORT_TABLESPACES parameter cannot be used in conjunction with the QUERY parameter.
- In RDBMS versions < 10.2.0.4 there was a 4K character-limit on the transportable_tablespaces parameter; a fix in 10.2.0.4 increased this to 32K. The compatibility parameter must be at least 10.2.0.4 or higher to implement this higher limit. (See Document 1131484.1 ExpdpTransportable Tablespace Fails With ORA-39071 for further details.)

Reference : Oracle Database Utilities - Chapter 3 Data Pump Import
Affected Version : Greater than 11.0.0, except where noted.

```
expdp <system/password> directory=<> dumpfile=<dump file name> logfile=<>
transport_tablespaces = <list of TSs>
```

Output

```
impdp system/password > directory=<> dumpfile=expdat.dmp directory=<> logfile=<>
transport_datafiles=<> exclude=table_statistics,index_statistics
```

Output

Import traditional

```
exp \'/ as sysdba\'
transport_tablespace=y
tablespaces=''
statistics=none
file=
log=

imp \'/ as sysdba\'
transport_tablespace=y
tablespaces=''
file=
log=
datafiles=
```

Validate the transported data on target

RMAN validate

```
RMAN> validate tablespace TS1, TS2 check logical;
```

Output

Compile invalid objects

```
declare
  parallel pls_integer:= 20;
begin
   utl_recomp.recomp_parallel(parallel);
end;
/
```

Check invalid objects

```
select count(*) from dba_objects where owner=’’;

select owner,object_type,object_name from dba_objects where owner=’’ and status=’INVALID’;
select owner,object_type,count(*) from dba_objects where owner=’’ group by
owner,object_type order by object_type;
```

	source	target
Owner Invalid objects		
Invalid objects summary		
Objects summary		

Transport tablespace check

```
select t.name "Tablespace", f.file# "File#", f.name "Filename", f.status "Status"
from v$datafile f, v$tablespace t
where f.ts#=t.ts# and t.name like 'TSPACE%'
order by 1, 2;
```

TTS verify check

```
sqlplus system/<> @tts_verify.sql
```

Sequence verify

```
col sequence_owner for a10
col sequence_name for a25
select a.sequence_owner,a.sequence_name,a.last_number,b.last_number from dba_sequences
a,dba_sequences@ttslink b
where a.sequence_owner=b.sequence_owner
and a.sequence_name=b.sequence_name
order by a.sequence_owner,a.sequence_name;
```

Put the tablespace(s) in read write on target

```
alter tablespace TS1 read write;
alter tablespace TS2 read write;
```

Output

Post health check

```
select t.name "Tablespace", f.file# "File#", f.name "Filename", f.status "Status"
from v$datafile f, v$tablespace t
where f.ts#=t.ts# and t.name like 'TSPACE%'
order by 1, 2;

select * from v$recover_file;
```

Option	Description
-S prepare source for transfer	-S option is used only when Prepare phase method is dbms_file_transfer. Prepare step is run once on the source system during Phase 2A with the environment (ORACLE_HOME and ORACLE_SID) set to the source database. This step creates files xttnewdatafiles.txt and getfile.sql.
-G get datafiles from source	-G option is used only when Prepare phase method is dbms_file_transfer. Get datafiles step is run once on the destination system during Phase 2A with the environment (ORACLE_HOME and ORACLE_SID) set to the destination database. The -S option must be run beforehand and files xttnewdatafiles.txt and getfile.sql transferred to the destination system. This option connects to the destination database and runs script getfile.sql. getfile.sql invokes dbms_file_transfer.get_file() subprogram for each datafile to transfer it from the source database directory object (defined by parameter srcdir) to the destination database directory object (defined by parameter dstdir) over a database link (defined by parameter srclink).
-p prepare source for backup	-p option is used only when Prepare phase method is RMAN backup. Prepare step is run once on the source system during Phase 2B with the environment (ORACLE_HOME and ORACLE_SID) set to the source database. This step connects to the source database and runs the xttpreparesrc.sql script once for each tablespace to be transported, as configured in xtt.properties. xttpreparesrc.sql does the following: 1. Verifies the tablespace is online, in READ WRITE mode, and contains no offline datafiles. 2. Identifies the SCN that will be used for the first iteration of the incremental backup step and writes it into file $TMPDIR/xttplan.txt. 3. Creates the initial datafile copies on the destination system in the location specified by the parameter dfcopydir set in xtt.properties. These datafile copies must be transferred manually to the destination system. 4. Creates RMAN script $TMPDIR/rmanconvert.cmd that will be used to convert the datafile copies to the required endian format on the destination system.
-c convert datafiles	-c option is used only when Prepare phase method is RMAN backup. Convert datafiles step is run once on the destination system during Phase 2B with the environment (ORACLE_HOME and ORACLE_SID) set to the destination database. This step uses the rmanconvert.cmd file created in the Prepare step to convert the datafile copies to the proper endian format. Converted datafile copies are written on the destination system to the location specified by the parameter storageondest set in xtt.properties.
-i create incremental	Create incremental step is run one or more times on the source system with the environment (ORACLE_HOME and ORACLE_SID) set to the source database. This step reads the SCNs listed in $TMPDIR/xttplan.txt and generates an incremental backup that will be used to roll forward the datafile copies on the destination system.
-r rollforward datafiles	Rollforward datafiles step is run once for every incremental backup created with the environment (ORACLE_HOME and ORACLE_SID) set to the destination database. This step connects to the incremental convert instance using the parameters cnvinst_home and cnvinst_sid, converts the incremental backup pieces created by the Create Incremental step, then connects to the destination database and rolls forward the datafile copies by applying the incremental for each tablespace being transported.
-s determine new FROM_SCN	Determine new FROM_SCN step is run one or more times with the environment (ORACLE_HOME and ORACLE_SID) set to the source database. This step calculates the next FROM_SCN, records it in the file xttplan.txt, then uses that SCN when the next incremental backup is created in step 3.1. It reports the mapping of the new

	FROM_SCN to wall clock time to indicate how far behind the changes in the next incremental backup will be.
-e generate Data Pump TTS command	Generate Data Pump TTS command step is run once on the destination system with the environment (ORACLE_HOME and ORACLE_SID) set to the destination database. This step creates the template of a Data Pump Import command that uses a network_link to import metadata of objects that are in the tablespaces being transported.
-d debug	-d option enables debug mode for xttdriver.pl and RMAN commands it executes. Debug mode can also be enabled by setting environment variable XTTDEBUG=1.

Create additional database link (optional)

```
create public database link to linka connect to system identified by Welcome1 using
'(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL = TCP)(HOST =
192.168.10.100)(PORT = 1521)))(CONNECT_DATA = (SERVICE_NAME = ptgdb)))';
```

Gather schema statistics on target

```
select dbms_stats.get_prefs('CONCURRENT') from dual;

begin
dbms_stats.set_global_prefs('CONCURRENT','TRUE');
end;
/
exec dbms_stats.gather_schema_stats(ownname => '<>',degree => 6);
```

Backup whole database

```
rman target /
backup database;
```

Phase 5 – Cleanup

Cleanup all files created for this process

Cleanup staging area if not done already

XTTS Reference

xttdriver.pl options

xtt.properties parameters

Parameter	Description	Example Setting

tablespaces	Comma-separated list of tablespaces to transport from source database to destination database. Must be a single line, any subsequent lines will not be read.	tablespaces=TS1,TS2
platformid	Source database platform id, obtained from V$DATABASE.PLATFORM_ID.	platformid=2

srcdir	Directory object in the source database that defines where the source datafiles currently reside. Multiple locations can be used separated by ",". The srcdir to dstdir mapping can either be N:1 or N:N. i.e. there can be multiple source directories and the files will be written to a single destination directory, or files from a particular source directory can be written to a particular destination directory.
	This parameter is used only when Prepare phase method is dbms_file_transfer.
		srcdir=SOURCEDIR
		srcdir=SRC1,SRC2

dstdir	Directory object in the destination database that defines where the destination datafiles will be created. If multiple source directories are used (srcdir), then multiple destinations can be defined so a particular source directory is written to a particular destination directory.
	This parameter is used only when Prepare phase method is dbms_file_transfer.
		dstdir=DESTDIR
		dstdir=DST1,DST2

srclink	Database link in the destination database that refers to the source database. Datafiles will be transferred over this database link using dbms_file_transfer. This parameter is used only when Prepare phase method is dbms_file_transfer.	srclink=TTSLINK
dfcopydir	Location on the source system where datafile copies are created during the "-p prepare" step. This location must have sufficient free space to hold copies of all datafiles being transported. This location may be an NFS-mounted filesystem that is shared with the destination system, in which case it should reference the same NFS location as the stageondest parameter for the destination system. See Note 359515.1 for mount option guidelines. This parameter is used only when Prepare phase method is RMAN backup.	dfcopydir=/stage_source
backupformat	Location on the source system where incremental backups are created. This location must have sufficient free space to hold the incremental backups created for one iteration through the process documented above. This location may be an NFS-mounted filesystem that is shared with the destination system, in which case it should reference the same NFS location as the stageondest parameter for the destination system.	backupformat=/stage_source
stageondest	Location on the destination system where datafile copies are placed by the user when they are transferred manually from the source system.	stageondest=/stage_dest

	This location must have sufficient free space to hold copies of all datafiles being transported. This is also the location from where datafiles copies and incremental backups are read when they are converted in the "-c conversion of datafiles" and "-r roll forward datafiles" steps. This location may be a DBFS-mounted filesystem. This location may be an NFS-mounted filesystem that is shared with the source system, in which case it should reference the same NFS location as the dfcopydir and backupformat parameters for the source system. See Note 359515.1 for mount option guidelines.	

storageondest	Location on the destination system where the converted datafile copies will be written during the "-c conversion of datafiles" step.
	This location must have sufficient free space to permanently hold the datafiles that are transported.
	This is the final location of the datafiles where they will be used by the destination database.
	This parameter is used only when Prepare phase method is RMAN backup.
		storageondest=+DATA
		- or -
		storageondest=/oradata/prod/%U

backupondest	Location on the destination system where converted incremental backups on the destination system will be written during the "-r roll forward datafiles" step. This location must have sufficient free space to hold the incremental backups created for one iteration through the process documented above. NOTE: If this is set to an ASM location then define properties asm_home and asm_sid below. If this is set to a file system location, then comment out asm_home and asm_sid parameters below.	backupondest=+RECO
cnvinst_home	Only set this parameter if a separate incremental convert home is in use. ORACLE_HOME of the incremental convert instance that runs on the destination system.	cnvinst_home=/u01/app/oracle/product/11.2.0.4/xtt_home
cnvinst_sid	Only set this parameter if a separate incremental convert home is in use. ORACLE_SID of the incremental convert instance that runs on the destination system.	cnvinst_sid=xtt
asm_home	ORACLE_HOME for the ASM instance that runs on the destination system. NOTE: If backupondest is set to a file system location, then comment out both asm_home and asm_sid.	asm_home=/u01/app/11.2.0.4/grid

asm_sid	ORACLE_SID for the ASM instance that runs on the destination system.	asm_sid=+ASM1
parallel	Defines the degree of parallelism set in the RMAN CONVERT command file rmanconvert.cmd. This file is created during the prepare step and used by RMAN in the convert datafiles step to convert the datafile copies on the destination system. If this parameter is unset, xttdriver.pl uses parallel=8. NOTE: RMAN parallelism used for the datafile copies created in the RMAN Backup prepare phase and the incremental backup created in the rollforward phase is controlled by the RMAN configuration on the source system. It is not controlled by this parameter.	parallel=3
rollparallel	Defines the level of parallelism for the -r roll forward operation.	rollparallel=2
getfileparallel	Defines the level of parallelism for the -G operation Default value is 1. Maximum supported value is 8.	getfileparallel=4

```
## Reduce Transportable Tablespace Downtime using Incremental Backups
## (Doc ID 1389592.1)
## (Doc ID 2005729.1)
##
## Properties file for xttdriver.pl
##
## Properties to set are the following:
##   tablespaces
##   platformid
##   srcdir
##   dstdir
##   srclink
##   dfcopydir
##   backupformat
##   stageondest
##   storageondest
##   backupondest
##   cnvinst_home
##   cnvinst_sid
##   asm_home
##   asm_sid
##   parallel
##   rollparallel
##   getfileparallel
##   metatransfer
##   destuser
##   desthost
##   desttmpdir
##   allowstandby
```

```
##
## See documentation below and My Oracle Support Notes 1389592.1(11g) and 2005729.1 (12c) for
details.
##


## Tablespaces to transport
## ========================
##
## tablespaces
## -----------
## Comma separated list of tablespaces to transport from source database
## to destination database.
## Specify tablespace names in CAPITAL letters.
tablespaces=TS1,TS2


## Source database platform ID
## ===========================
##
## platformid
## ----------
## Source database platform id, obtained from V$DATABASE.PLATFORM_ID
platformid=13


## Parameters required for Prepare Phase method dbms_file_transfer
## ===============================================================
##
## srcdir
## ------
## Directory object in the source database that defines where the source
## datafiles currently reside.
## Feb 2015: Ver2: We support multiple SOURCEDIR's.
## NOTE: Number of entries in srcir and dstdir should match.
## The srcdir to dstdir mapping can either be N:1 or N:N i.e. there can be
## multiple source dirs and the files will be written to a single destination
## directory, or files from a particular source dir can be written to a
## particular destination directory
## Example [N:1, allowed]
## ======================
## srcdir=SRC1,SRC2
## dstdir=DST
##
## In this case the files from SRC1, SRC2 will be written to DST
##
```

```
## Example [N:N, allowed]
## ======================
## srcdir=SRC1,SRC2
## dstdir=DST1,DST2
##
## In this case the files from SRC1 will be written to DST1, SRC2 to DST2.
##
## Example [N:M, not allowed]
## ==========================
## srcdir=SRC1,SRC2,SRC3
## dstdir=DST1,DST2
##
## This is not allowed and will result in error.
#srcdir=SOURCEDIR1,SOURCEDIR2

## dstdir
## ------
## Directory object in the destination database that defines where the
## destination datafiles will be created.
## Feb 2015: Ver2: We support multiple DESTDIR's.
## SOURCEDIR1 will map to DESTDIR1 and SOURCEDIR2 to DESTDIR2 and so on
## Refer to above parameter for more examples
#dstdir=DESTDIR1,DESTDIR2

## srclink
## -------
## Database link in the destination database that refers to the source
## database.  Datafiles will be transferred over this database link using
## dbms_file_transfer.
srclink=TTSLINK



## Source system file locations
## ============================
##
## dfcopydir
## ---------
## This parameter is used only when Prepare phase method is RMAN backup.
##
## Location where datafile copies are created during the "-p prepare" step.
## This location must have sufficient free space to hold copies of all
## datafiles being transported.
##
## This location may be an NFS-mounted filesystem that is shared with the
## destination system, in which case it should reference the same NFS location
## as the stageondest property for the destination system.
dfcopydir=/stage_source

## backupformat
## ------------
## Location where incremental backups are created.
##
## This location may be an NFS-mounted filesystem that is shared with the
## destination system, in which case it should reference the same NFS location
## as the stageondest property for the destination system.
backupformat=/stage_source


## Destination system file locations
## ==================================
##
## stageondest
## -----------
## Location where datafile copies are placed by the user when they are
## transferred manually from the souce system.  This location must have
## sufficient free space to hold copies of all datafiles being transported.
##
## This is also the location from where datafiles copies and incremental
## backups are read when they are converted in the "-c conversion of datafiles"
## and "-r roll forward datafiles" steps.
##
## This location may be a DBFS-mounted filesystem.
##
## This location may be an NFS-mounted filesystem that is shared with the
## source system, in which case it should reference the same NFS location
## as the dfcopydir and backupformat properties for the source system.
stageondest=/stage_dest

## storageondest
## -------------
## This parameter is used only when Prepare phase method is RMAN backup.
##
## Location where the converted datafile copies will be written during the
## "-c conversion of datafiles" step.  This is the final location of the
## datafiles where they will be used by the destination database.
storageondest=+DATA

## backupondest
## ------------
## Location where converted incremental backups on the destination system
## will be written during the "-r roll forward datafiles" step.
```

```
##
## NOTE: If this is set to an ASM location then define properties
##       asm_home and asm_sid below.  If this is set to a file system
##       location, then comment out asm_home and asm_sid below
backupondest=+RECO


## Database home and SID settings for destination system instances
## ============================================================
##
## cnvinst_home, cnvinst_sid
## -------------------------
## Database home and SID of the incremental convert instance that
## runs on the destination system.
##
## Only set these parameters if a separate incremental convert home is in use.
#cnvinst_home=/u01/app/oracle/product/11.2.0.4/xtt_home
#cnvinst_sid=xtt

## asm_home, asm_sid
## -----------------
## Grid home and SID for the ASM instance that runs on the destination
## system.
##
## NOTE: If backupondest is set to a file system location, then comment out
##       both asm_home and asm_sid.
#asm_home=/u01/app/11.2.0.4/grid
#asm_sid=+ASM1


## Parallel parameters
## ===================
##
## parallel
## --------
## Parallel defines the channel parallelism used in copying (prepare phase),
## converting.
##
## Note: Incremental backup creation parallelism is defined by RMAN
## configuration for DEVICE TYPE DISK PARALLELISM.
##
## If undefined, default value is 8.
parallel=3

## rollparallel
## ------------
## Defines the level of parallelism for the -r roll forward operation.
##
## If undefined, default value is 0 (serial roll forward).
rollparallel=2

## getfileparallel
## ---------------
## Defines the level of parallelism for the -G operation
##
## If undefined, default value is 1. Max value supported is 8.
## This will be enhanced in the future to support more than 8
## depending on the destination system resources.
getfileparallel=4

## metatransfer
## ---------------
## If passwordless ssh is enabled between the source and the destination, the
## script can automatically transfer the temporary files and the backups from
## source to destination. Other parameters like desthost, desttmpdir needs to
## be defined for this to work. destuser is optional
## metatransfer=1

## destuser
## ---------
## The username that will be used for copying the files from source to dest
## using scp. This is optional
## destuser=username

## desthost
## --------
## This will be the name of the destination host.
## desthost=machinename

## desttmpdir
## ---------------
## This should be defined to same directory as TMPDIR for getting the
## temporary files. The incremental backups will be copied to directory pointed
## by stageondest parameter.
## desttmpdir=/tmp

## dumpdir
## ---------
## The directory in which the dump file be restored to. If this is not specified
## then TMPDIR is used.
## dumpdir=/tmp
## using scp. This is optional
## destuser=username

## allowstandby
## ---------
## This will allow the script to be run from standby database.
## allowstandby=1

## END
```

Scripts Reference

tts_verify.sql

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

```
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
Platform Migration Using Transportable Tablespaces: Oracle Database 11g Release 1 Page 30
Maximum Availability Architecture
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
(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$skip_support
where action=0) and status='INVALID'
group by owner, object_type ) r
, ( select owner, object_type, count(owner) local_cnt
from dba_objects
where owner not in
(select name
from system.logstdby$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
```

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
```

Reference

https://mikedietrichde.com/2017/10/05/oow-2017-slides-download-migrate-100-tb-databases-less-one-day/

http://docs.oracle.com/database/121/ADMIN/transport.htm#ADMIN13721

http://docs.oracle.com/database/122/ADMIN/transporting-data.htm#ADMIN13721

NOTE:1079563.1 - RMAN DUPLICATE/RESTORE/RECOVER Mixed Platform Support

NOTE:413484.1 - Data Guard Support for Heterogeneous Primary and Physical Standbys in Same Data Guard Configuration

(Doc ID 1902618.1) Migrate database to Exadata with DBMS_FILE_TRANSFER (Doc ID 371556.1)How to Migrate to different Endian Platform Using Transportable Tablespaces With RMAN

(Doc ID 1389592.1) 11G - Reduce Transportable Tablespace Downtime using Cross Platform Incremental Backup

(Doc ID 1454872.1) Transportable Tablespace (TTS) Restrictions and Limitations: Details, Reference, and Version Where Applicable

(Doc ID 2005729.1) 12C - Reduce Transportable Tablespace Downtime using Cross Platform Incremental Backup

Note:1166564.1 Master Note for Transportable Tablespaces Common Questions and Issues

Note:1454872.1 Transportable Tablespace Restrictions and Limitations: Details, Reference, and Version Where Applicable

For TTS White Papers see the MAA webpage:
http://www.oracle.com/technetwork/database/features/availability/oracle-database-maa-best-practices-155386.html

Database Upgrades using TTS:
http://www.oracle.com/technetwork/database/features/availability/maa-wp-11g-upgradetts-132620.pdf

Platform Migration using Transportable Database (RMAN):
http://www.oracle.com/technetwork/database/features/availability/maa-wp-10gr2-platformmigrationtdb-131164.pdf

Customer example: Amadeus Customer Case
http://www.oracle.com/technetwork/database/features/availability/s281209-amadeus-130978.pdf