Migrating bulky Oracle databases to AWS for cross-platform environments

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Migrating bulky Oracle databases to AWS for cross-platform environments

Table of Contents

Introduction **1**
Overview **2**
Phase 1 - Prepare phase 3
Phase 2 - Roll-forward phase 3
Phase 3 - Transport phase 4
Migration phases **5**
Phase 0: Setup phase 5
Phase 1: Prepare phase 6
Step 1: Take full (level=0) tablespace backups on the source system 6
Step 2: Transfer full backup copies to the destination system 7
Step 3: Restore full backup copies and convert data files 10
Phase 2: Roll-forward phase 10
Step 1: Take incremental backups in parallel 11
Step 2: Transfer the incremental backups and res.txt file to the destination system 11
Step 3: Convert and apply incremental backups 11
Phase 3: Transport phase 14
Step 1: Make the tablespaces in the source database read only 14
Step 2: Create the final incremental backup 14
Step 3: Export the metadata 14
Step 4: Transfer the files and apply the final incremental backup 15
Step 5: Import the object metadata 16
Phase 4: Validate 17
Step 1: Check tablespaces for corruption 17
Step 2. Make all transported tablespaces in the destination database read/write 17
Conclusion **19**
Document history **20**

Migrating bulky Oracle databases to AWS for cross-platform environments

Incheol Roh, Amazon Web Services

When migrating an Oracle database larger than 100 TB from on premises to the Amazon Web Services (AWS) Cloud, migration downtime is a serious factor. In particular, if the system is a mission-critical system, such as global enterprise resource planning (ERP) or a manufacturing execution system (MES), you want to reduce the downtime as much as possible for business continuity.

There are many ways to migrate Oracle databases between systems that have different endian formats, as mentioned in Strategies for Migrating Oracle Databases to AWS. One of these approaches is Oracle cross-platform transportable tablespaces (XTTS), which you can use to reduce the migration downtime from days to hours.

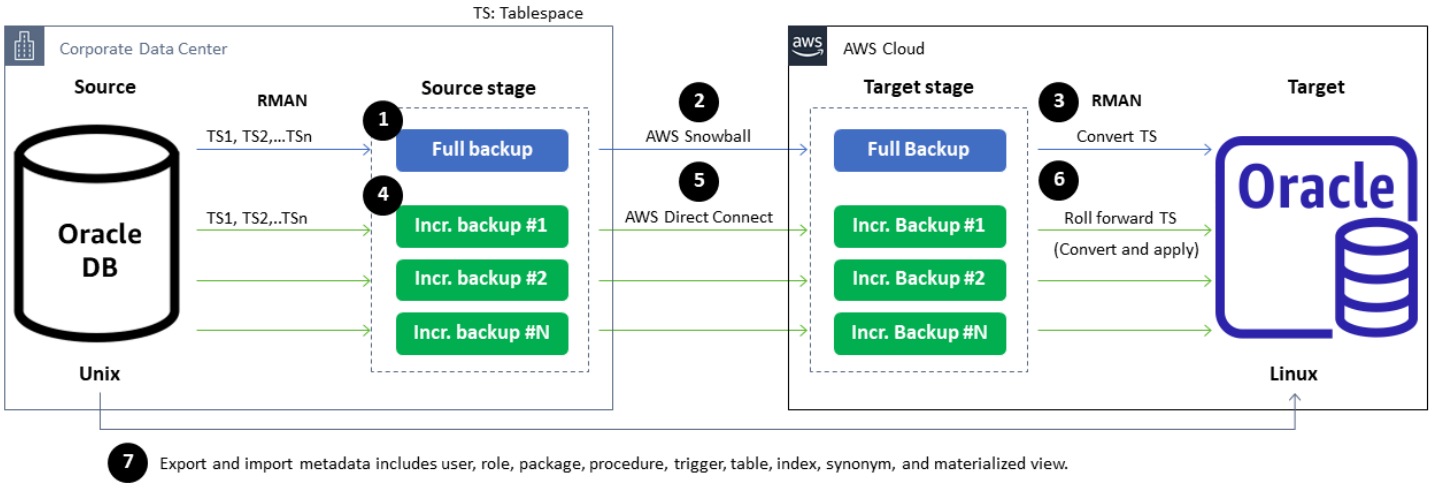
Combining Oracle XTTS with Oracle Recovery Manager (RMAN) incremental backups can significantly reduce the amount of downtime required to move data between platforms running different endian formats.

This guide introduces how to use AWS Snowball, AWS Direct Connect, and Amazon FSx for Lustre with Oracle XTTS with RMAN incremental backups . The goal of this approach is to minimize migration downtime in environments that have very large datasets.

Overview

This is the conceptual process of migrating Oracle databases to AWS using Oracle XTTS and RMAN incremental backups with Snowball, Direct Connect, and FSx for Lustre.

The following diagram shows the high-level migration steps for an Oracle database across different endian formats.



1. Make a full backup of all the tablespaces.
2. Use Snowball to move the backup from the source stage to the target stage.
3. Convert the tablespaces to the target database.
4. Make incremental backups.
5. Use Direct Connect to transfer incremental backups from the source stage to the target stage.
6. Roll forward the incremental backups, converting them and applying them to the target database.
7. Export and import metadata for all tablespaces being transported.

Before cutover, you can minimize downtime by doing the following:

- Exporting and importing the metadata of nonsegment-based objects, including USER, PACKAGE_SPEC, PACKAGE_BODY, PROCEDURE, and FUNCTION
- Increasing parallelism for full backup and incremental backup
- Converting data files

- Rolling forward backups during migration

The Oracle document Reduce Transportable Tablespace Downtime using Cross Platform Incremental Backup (2471245.1) explains how to use Oracle XTTS with RMAN incremental backups. The document also includes details on requirements and recommendations. The document doesn't describe how to migrate an Oracle database from an on-premises environment to Oracle on AWS or how to parallelize each migration step to minimize downtime.

This guide provides a way to parallelize the phases, minimizing migration downtime in mission-critical system environments with tremendously large data sizes.

Following an initial setup phase, the high-level steps for using Oracle XTTS with RMAN incremental backups include the following phases.

Phase 1 - Prepare phase

The prepare phase consists of the following steps.

1. An initial full backup (level=0) of the tablespaces is taken on the source database to the source stage, which is NAS storage.
2. The backup copies are transferred using Snowball to the target stage, which is FSx for Lustre integrated with Amazon Simple Storage Service (Amazon S3).
3. Backup tablespaces are restored and converted to the target database with little-endian format.

The steps in this phase are run only once during migration. The data being transported are fully accessible in the source database during this phase.

Phase 2 - Roll-forward phase

The roll-forward phase consists of the following steps.

1. An incremental backup is taken from the source database to the source stage.
2. Incremental backup copies are transferred to the target stage over Direct Connect.
3. Incremental backup copies are converted to the target database with little-endian format. The copies are then applied to the initial target database, which is called the roll-forward step.

You can run this phase multiple times. Each successive incremental backup should take less time and will bring the destination data file copies more current with the source database. As in phase 1, the source data being transported are fully accessible during this phase.

Phase 3 - Transport phase

The third phase includes the following steps.

1. The tablespaces being transported are changed to read only.
2. A final incremental backup is taken from the source database.
3. The metadata is exported.
4. Backups are transferred and applied to the destination.
5. Object metadata is imported.

At this point, the system change number (SCN) of the destination database is consistent with that of source database.

The metadata of transportable tablespaces are exported from the source database and imported into the destination database. The metadata includes information for the user, role, package, procedure, function, table, and index.

Finally, the tablespaces are made read/write for full access on the destination database from application.

Migration phases

This guide covers migrating both Oracle single instance and Oracle RAC as the source and target database. If the source and target is Oracle RAC, you can maximize the parallelism for each migration step.

Phase 0: Setup phase

1. Install Oracle database 12.1.0.2 or later on the target Amazon Elastic Compute Cloud (Amazon EC2) instance.
2. Verify the target database.

   You must verify whether the target database instance has the same time zone and character set as source database. Otherwise, this migration approach will fail.

   To check that the time zone version in both the source and target is same, run the following command.

```
SQL> select * from v$timezone_file;
FILENAME            VERSION    CON_ID
----------------------   -------------   ----------
timezlrg_14.dat        14              0
```

   To check that both the DB character set and the national character set are the same in the source and target, run the following command.

```
SQL> select * from nls_database_parameters
         where parameter in ('NLS_CHARACTERSET','NLS_NCHAR_CHARACTERSET');
PARAMETER                                VALUE
-----------------------------------------   --------------------
NLS_NCHAR_CHARACTERSET         UTF8
NLS_CHARACTERSET                    AL32UTF8
```

3. Set up the Oracle XTTS utility.

On the source system, download and extract the script file, rman_xttconvert_VER4.zip from Oracle document 2471245.1. Modify the parameters in the xtt.properties file with the specific configuration. Then copy xtt.properties and xttdriver.pl scripts to the destination system.

Phase 1: Prepare phase

During this phase, data files of the tablespace to be transported are backed up on the source system. Backup copies are transported to the destination system and restored by running the xttdriver.pl script.

Step 1: Take full (level=0) tablespace backups on the source system

To reduce the backup duration time, copy the xtt.properties and xttdriver.pl files to multiple directories. In each xtt.properties file under each directory, enter the name of the tablespace in the tablespaces parameter. Then, under each directory, run xttdriver.pl with the --backup option. This command transports all the tablespaces except for SYSTEM, SYSAUX, UNDO, and TEMP, which were created during installation of the target database.

For example, each xtt01~xtt04 directory has all xtt scripts (xtt.properties and xttdriver.pl) with different values for the tablespaces= parameter in the xtt.properties file. When editing the tablespaces parameter in each xtt.properties file, consider dividing the tablespace groups so that the total size of the data files in each tablespace group is similar.

In this guide, the example supposes that there are four tablespace groups. If the source database is Oracle single instance, you create all xtt01~04 directories. If the source database is Oracle RAC, you can create each xtt directory in each Oracle RAC server.

```
[oracle@erp expimp]$ ls
out  xtt01  xtt02  xtt03    xtt04
[oracle@erp out]$ ls
out01 out02 out03 out04
```

The following table shows examples of the tablespaces= parameter in the xtt.properties files.

xtt01	tablespaces=APPS_TS_TX_DATA
xtt02	tablespaces=APPS_TS_TX_IDX

xtt03	tablespaces=APPS_TS_SUMMARY
xtt04	tablespaces=APPS_CALCLIP, APPS_OMO, APPS_TS_ARCHIVE, APPS_TS_DISCO, APPS_TS_DISCO_OLAP, APPS_TS_INTERFACE, APPS_TS_MEDIA, APPS_TS_NOLOGGING, APPS_TS_QUEUES, APPS_TS_SEED…

To prevent any existing data file backup copies from being deleted while xttdriver.pl is running in parallel, comment out the following line in xttdriver.pl.

```
   if (@present)
   {
      ## Try deleting any existing copied datafiles
##Comment out it for executing rman command in parallel by AWS
##        system("\\rm -f $dfcopydir/*.tf");
      sleep 5;
   }
```

Use xttdriver.pl to make the RMAN backup of the source system.

If the source system is Oracle RAC, it can be run on each Oracle RAC instance simultaneously.

The output of xttdriver.pl is stored in the TMPDIR environment variable. Run each xttdriver.pl command with the --backup option, and use the --debug option to turn on debug mode.

```
cd /u01/oracle/expimp/xtt<nn>
export TMPDIR=/u01/oracle/expimp/out/out<nn>
$ORACLE_HOME/perl/bin/perl xttdriver.pl --backup --debug 3
```

In this example, <*nn*> represents the tablespace groups. If there are four tablespace groups, <*nn*> becomes 01, 02, 03, and 04.

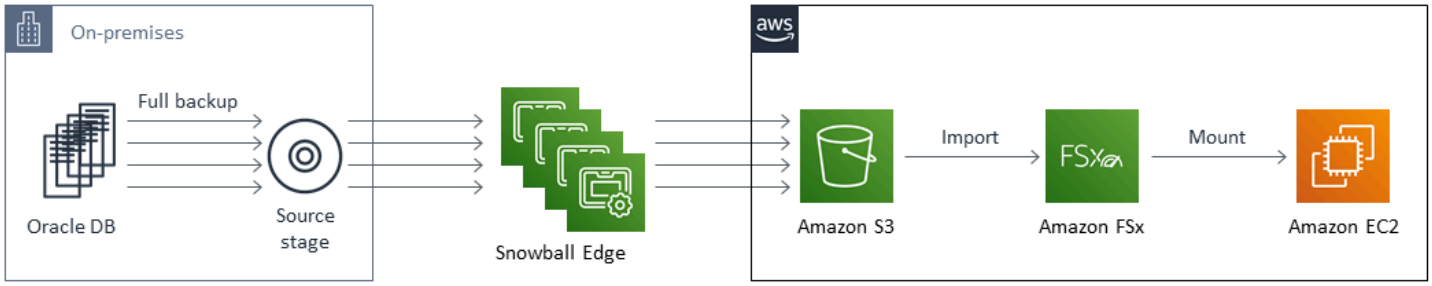
Step 2: Transfer full backup copies to the destination system

Option 1 - Using AWS Snowball Edge

Path: Source database -> source stage -> AWS Snowball Edge -> Amazon S3 -> Amazon FSx for Lustre

> **Note**
>
> AWS Snowball Edge is no longer available to new customers. New customers should explore AWS DataSync for online transfers, AWS Data Transfer Terminal for secure physical transfers, or AWS Partner solutions. For edge computing, explore AWS Outposts.

The following diagram shows the transfer of a full backup using Snowball Edge.



The Snowball Edge is a rugged physical storage and computing device that you can use to move large-scale data to AWS. Snowball Edge helps overcome challenges that you might encounter with large-scale data transfers, including long transfer times, a lack of usable bandwidth, and security concerns.

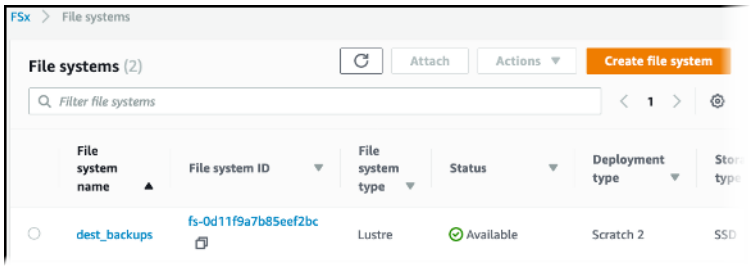
The entire data file backup copies are transferred into Amazon S3 by the Snowball Edge. If the source system size is over 100 TB, you must use multiple Snowball Edge devices to reduce the duration of the transfer.

Amazon FSx for Lustre is deeply integrated with Amazon S3. You can create an FSx for Lustre file system that is linked to your S3 bucket in minutes. When linked to the Amazon S3 data repository, FSx for Lustre file systems transparently present Amazon S3 objects as files. In the real environment, the performance of FSx for Lustre depends on its storage capacity, the size of each file, and how well files are distributed into the file system. When FSx for Lustre is used as target-stage storage, you don't need to restore tablespace backup copies into Amazon Elastic Block Store (Amazon EBS) or Amazon Elastic File System from Amazon S3.

1. Import the S3 bucket to Amazon FSx for Lustre.

   Use Snowball Edge to transfer and store the source stage area (for example, src_backups) into an S3 bucket (for example, s3-src-backups).

Create the FSx for Lustre file system (for example, dest_backups) as the destination stage area for the data file backup copies.



Install the open-source Lustre client on the destination system (Amazon EC2). For more information, see the Amazon FSx for Lustre User Guide.

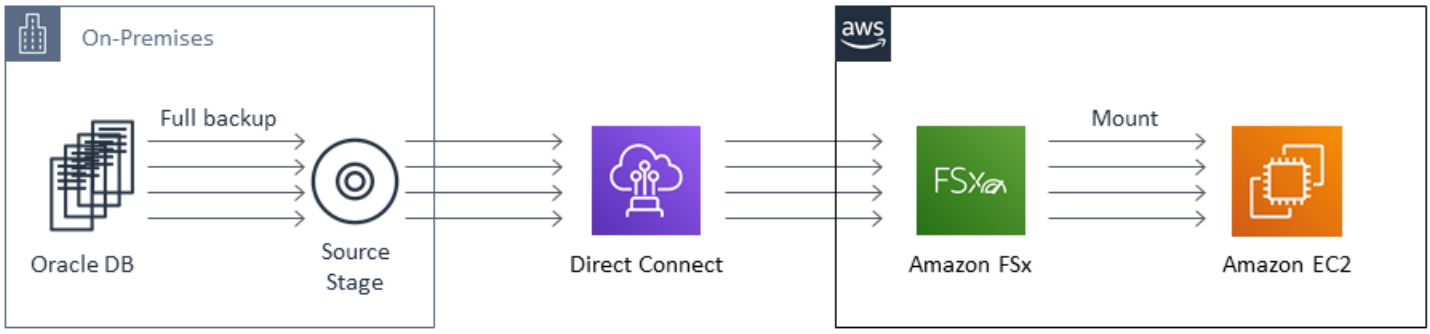
After the Lustre client is installed, mount the FSx for Lustre file system to the destination system (Amazon EC2).

2. Transfer res.txt from $TMPDIR on the source to $TMPDIR on the destination.

Option 2 - Using AWS Direct Connect

Path: Source database -> source stage -> AWS Direct Connect -> Amazon FSx for Lustre

The following diagram shows the transfer of a full backup by using AWS Direct Connect.



Direct Connect gives you the ability to create private network connections between your data center, office, or colocation environment and AWS. These private network connections reduce network costs, increase throughput, and deliver a consistent experience.

You can directly transfer data file backup copies from the source system to the destination system (Amazon EC2) where the Amazon FSx for Lustre file system is mounted. This option is faster than the Snowball Edge option if you can accommodate the high bandwidth of Direct Connect.

After data file backup copies are transferred by Snowball or Direct Connect, you can see them in the FSx for Lustre file system in the target stage area.

Step 3: Restore full backup copies and convert data files

Restore full backup copies and convert data files to destination system endian format. To reduce the restore and convert duration, run the xttdriver.pl command with the --restore option for each tablespace group.

```
cd /u01/oracle/expimp/xtt<nn>
export TMPDIR=/u01/oracle/expimp/out/out<nn>
$ORACLE_HOME/perl/bin/perl xttdriver.pl --restore --debug 3
```

After the step is complete, data files are placed in the defined *dest_datafile_location* in the xtt.properties file on the destination system.

Phase 2: Roll-forward phase

You can repeat the roll-forward phase as many times as necessary to catch the destination data file up to source database.

The roll-forward phase consists of the following steps.

1. Create an incremental backup from the source database.
2. Transfer the backup to the destination system.
3. Convert the backup to the destination system endian format.
4. Apply the backup to the converted destination data file copies to roll them forward.

Each successive incremental backup should take less time and will bring the destination data file copies more current with the source database.

Step 1: Take incremental backups in parallel

Take incremental backups of the tablespace groups being transported on the source system in parallel.

This step creates incremental backups for all tablespaces= that are listed in the xtt.properties file.

If you can activate the **BLOCK CHANGE TRACKING** feature on the source system, you can greatly reduce the time of the incremental backup.

The following command recognizes the next SCN after the full backup automatically.

```
cd /u01/oracle/expimp/xtt<nn>
export TMPDIR=/u01/oracle/expimp/out/out<nn>
$ORACLE_HOME/perl/bin/perl xttdriver.pl --backup --debug 3
```

Step 2: Transfer the incremental backups and res.txt file to the destination system

If you have enough bandwidth, you can reduce the transferring duration by using Direct Connect.

Transfer the incremental backups between *src_scratch_location* and *dest_scratch_location*. Transfer the res.txt file between $TMPDIR on the source system and $TMPDIR on the destination system.

If you take multiple incremental backups, the res.txt file must be copied after the last incremental backup before it can be applied on destination system.

```
[source]$ scp $TMPDIR/res.txt oracle@[dest]:/u01/oracle/expimp/out/out<nn>
[source]$ scp <src_scratch_location>/* oracle@[dest]:<dest_scratch_location>
```

Step 3: Convert and apply incremental backups

Convert incremental backups to the destination system endian format and apply incremental backups to the destination data file copies on the destination system.

In this guide, suppose that tablespace groups are four. Run each xttdriver.pl command with the --restore option for each tablespace group.

In this step, the Oracle XTTS utility takes the target database to be restarted. To run the command in parallel, you must use the following customization in the Perl script, xttdriver.pl.

1. Comment out the following lines:
   - Line 4867: my $outputstart = `sqlplus -L -s \"/ as sysdba\" \@xttstartupnomount.sql`;
   - Line 4868: checkError ("Error in executing xttstartupnomount.sql", $outputstart);
   - Line 4992: my $outputstart = `sqlplus -L -s \"/ as sysdba\" \@xttdbopen.sql`;\
2. Make the target DB in NOMOUNTstatus.

```
sqlplus / as sysdba @xttstartupnomount.sql
```

3. Perform a roll forward of the incremental backups in parallel.

```
cd /u01/oracle/expimp/xtt<nn>
export TMPDIR=/u01/oracle/expimp/out/out<nn>
$ORACLE_HOME/perl/bin/perl xttdriver.pl --restore --debug 3
```

4. After rolling forward all incremental backups, set the target DB in OPENstatus.

```
sqlplus / as sysdba @xttdbopen.sql
```

At this point, you can repeat phase 2 until the SCNs on the source and destination databases get close enough. You must decide to whether go forward to phase 3 or repeat phase 2, depending on given target downtime.

To reduce the downtime during phase 3 (the transport phase), you can export and import the metadata of nonsegment-based objects, such as USER, PACKAGE, PROCEDURE, and FUNCTION, before you set the source database as READ ONLY.

If the number of database objects in the source database is extremely large (hundreds of thousands), exporting and importing the metadata will take several hours. In this case, consider exporting metadata.

Exporting nonsegment-based objects is run in read/write on the source system. Then you must import the objects to the destination system before the source system is set to READ ONLY. At this time, you must keep the database source objects such as PACKAGE, PROCEDURE, and FUNCTION unchanged.

This is an example of the dump parameter file that is used to export metadata of nonsegment-based objects, including USER, PACKAGE_SPEC, PACKAGE_BODY, PROCEDURE, and FUNCTION.

```
directory=dmpdir
dumpfile=xttsmsc%U.dmp
full=y
filesize=1048576000
logfile=expmsc.log
metrics=y
exclude=TABLE,INDEX,CONSTRAINT,COMMENT,
MATERIALIZED_VIEW,MATERIALIZED_VIEW_LOG,SCHEMA_CALLOUT
```

Use the following command to export metadata.

```
SQL> create directory dmpdir as <location>;
expdp system/<system password> parfile=<parameter file>
```

This is an example of the dump parameter file to import metadata of nonsegment objects.

```
directory=dmpdir
dumpfile=xttsmsc%U.dmp
full=y
logfile=impmsc1.log
EXCLUDE=TABLESPACE, PROCOBJ, RLS_CONTEXT, RLS_GROUP, RLS_POLICY, TABLESPACE_QUOTA
metrics=y
remap_tablespace=
APPS_TS_ARCHIVE:SYSTEM,
APPS_TS_INTERFACE:SYSTEM,
APPS_TS_SEED:SYSTEM,
APPS_TS_SUMMARY:SYSTEM,
…..
```

At this time, there are no tablespaces except SYSTEM, SYSAUX, UNDO, and TEMP on the destination system. So you must remap all objects to be imported to the SYSTEM tablespace.

Use the following command to import metadata.

```
SQL> create directory dmpdir as <location>;
impdp system/<system password> parfile=<parameter file>
```

Now you can see the created objects on the destination database.

```
SQL> select object_type, count(*) from dba_objects group by object_type order by
 count(*) desc;
```

```
OBJECT_TYPE                    COUNT(*)
------------------------------ ----------
SYNONYM                           89327
PACKAGE                            55670
PACKAGE BODY                     54447
VIEW                                  41378
JAVA CLASS                         31978
SEQUENCE                          12766
…..
```

There are no tablespaces except SYSTEM,/span>, SYSAUX, UNDO, and TEMP. The USER object is created with USERS as the default tablespace. During phase 3, the transportable tablespaces will be created, and then the USER objects will be altered with the original default tablespaces.

Phase 3: Transport phase

During this phase, the source system becomes read only. The data files on the destination system are made consistent with the source system by applying a final incremental backup. Then, export the object metadata from the source system and import it into the destination system.

Step 1: Make the tablespaces in the source database read only

As SYSDBA, make all tablespaces being transferred READ ONLY on the source system.

To reduce downtime, you can run the following two steps simultaneously.

Step 2: Create the final incremental backup

On the source system, create the final incremental backup of the tablespaces being transferred.

```
cd /u01/oracle/expimp/xtt<nn>
export TMPDIR=/u01/oracle/expimp/out/out<nn>
$ORACLE_HOME/perl/bin/perl xttdriver.pl --backup --debug 3
```

This step returns an error, such as "ORA-20001: TABLESPACE(S) IS READONLY"; the error is expected, and you can safely ignore it.

Step 3: Export the metadata

Export the metadata of the transportable tablespaces from the source database.

This is an example of the parameter file for exporting the metadata of the transportable tablespaces.

```
directory=dmpdir
metrics=y
dumpfile=xttsmeta%U.dmp
filesize=1048576000
logfile=expxtts.log
transport_tablespaces=
APPS_TS_ARCHIVE,
APPS_TS_INTERFACE,
APPS_TS_MEDIA,
APPS_TS_NOLOGGING,
….
exclude=table_statistics,index_statistics
```

In addition, if the source system has many tables and indexes, you can save time during exporting by excluding their statistics. After you import the transportable tablespace, import the statistics to the destination system.

Before you run expdp, create a database directory where dump files are stored on the source system.

```
SQL> create directory dmpdir as <location>;
expdp system/<system password> parfile=<parameter file>
```

The following two steps are the final steps for cross-platform transportable tablespace with RMAN incremental backups. These steps must be performed sequentially.

Step 4: Transfer the files and apply the final incremental backup

Transfer the final incremental backup and export dump files to the destination system, convert, and apply the final incremental backup.

Use Direct Connect to transfer the final incremental backup copies and res.txt file to the destination. You can use VPN connectivity, but using Direct Connect will reduce the downtime significantly if it has enough bandwidth.

To restore the final incremental backup, on the destination system, run the following command, using the --restore option, for each tablespace group.

```
cd /u01/oracle/expimp/xtt<nn>
export TMPDIR=/u01/oracle/expimp/out/out<nn>
$ORACLE_HOME/perl/bin/perl xttdriver.pl --restore --debug 3
```

Step 5: Import the object metadata

Import the object metadata into the destination system by using Oracle Data Pump. Run the following command to get the list of data files for the transport_datafiles= parameter on the destination system.

```
cd /u01/oracle/expimp/xtt<nn>
export TMPDIR=/u01/oracle/expimp/out/out<nn>
$ORACLE_HOME/perl/bin/perl xttdriver.pl -e
```

Whenever you run the previous command, you get the xttplugin.txt file, which has the transport_datafiles= parameter. Merge transport_datafiles= in a single line from all the xttplugin.txt files, and put the data file lists into the transport_datafiles argument of the parameter file for the import metadata.

The following code snippet shows the parameter file for importing the transportable tablespace on the destination system.

```
directory=dmpdir
metrics=y
dumpfile=xttsmeta%U.dmp
logfile=impxtts.log
exclude=TYPE
transport_datafiles= '+EBSDATA/APPS_TS_TX_DATA_2.dbf','+EBSDATA/
APPS_TS_TX_DATA_11.dbf','+EBSDATA/APPS_TS_TX_DATA_22.dbf','+EBSDATA/
APPS_TS_TX_DATA_183.dbf','+EBSDATA/APPS_TS_TX_DATA_204.dbf','+EBSDATA/
APPS_TS_TX_DATA_219.dbf','+EBSDATA/APPS_TS_TX_DATA_227.dbf'…..
```

Before running impdp, create a database directory pointing to the location of the export dump files.

```
SQL> create directory dmpdir as <location>;
impdp system/<system password> parfile=<parameter file>
```

Phase 4: Validate

Step 1: Check tablespaces for corruption

In this step, the transported tablespaces are read only in the destination database. You can validate whether the tablespaces are valid or not by running the RMAN command as follows.

```
sqlplus / as sysdba;
set feedback off
set pagesize 0
set heading off
spool tbs_val.sql;
select 'validate tablespace '||tablespace_name||' check
 logical;' from dba_tablespaces where tablespace_name not in
 ('SYSTEM','SYSAUX','TEMP','USERS','UNDOTBS1','UNDOTBS2','UNDOTBS3','UNDOTBS3','UNDOTBS4');
spool off;
exit;


--Remove the first and last line in the spool file, tbs_val.sql
rman target /
RMAN> @tbs_val.sql
```

In addition, you can check the count of the objects, such as the table, index, synonym, view, and package objects.

Step 2. Make all transported tablespaces in the destination database read/write

```
sqlplus / as sysdba;
set feedback off
set pagesize 0
set heading off
spool tbs_rw.sql
select 'alter tablespace '||tablespace_name||' read write;' from dba_tablespaces where
 status='READ ONLY';
spool off;
exit;
--Remove the first and last line in the spool file, tbs_rw.sql
sqlplus / as sysdba;
```

```
SQL> @tbs_rw.sql
```

Conclusion

In this guide, you learned how to minimize downtime by using Oracle cross-platform transportable tablespaces and RMAN incremental backups with AWS Snowball, AWS Direct Connect, and Amazon FSx for Lustre.

When migrating an Oracle database from on premises to AWS, consider the following variables:

- The network bandwidth required by Direct Connect
- The capacity of FSx for Lustre
- The metadata size of the Oracle database
- The incremental backup size

Using the method recommended in this guide, when migrating from an 100 TB Oracle database running on a Unix system to AWS, you can reduce the downtime to around 10 hours.

Document history

The following table describes significant changes to this guide.

Change	Description	Date
Update	Changed mentions of *heterogeneous* in the *Introduct ion* and *Overview*.	July 14, 2021
Initial publication	—	March 2, 2021