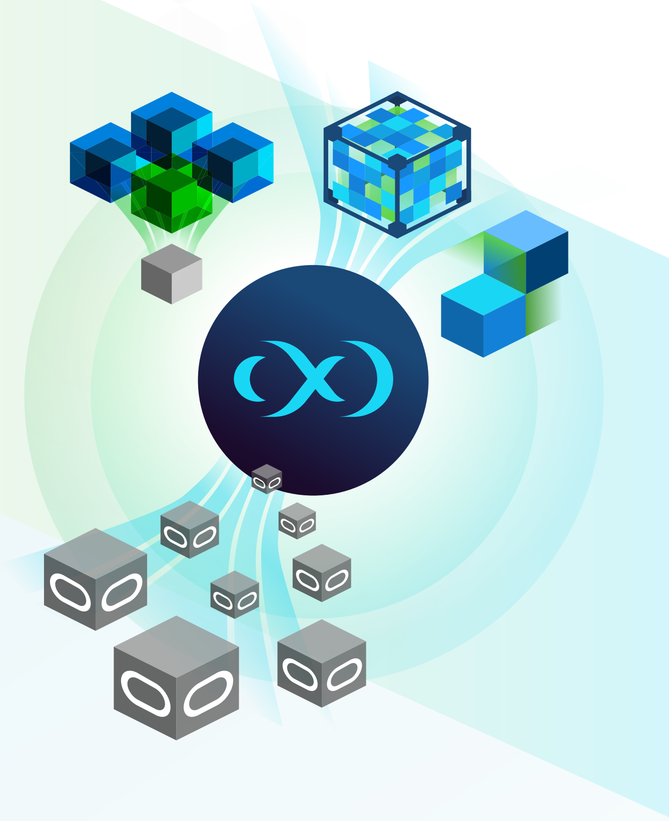
WHITE PAPER



Ingesting Oracle Backups into Delphix

Linking and ingesting data from Oracle databases, be it production or standby databases, traditionally involved Delphix taking a backup on the host where the database resides and using this backup as a way to create the initial copy of the database data on Delphix appliance. Subsequent changes in these production databases are captured with a combination of incremental backups and transaction logs (archived logs). With the introduction of the "Staging Push" feature, starting with the 7.0.0.0 release of the Delphix appliance, users can ingest their backups into Delphix without the need for Delphix initiated backups or even Delphix access to the source Database.

Current Ingestion Process

Ingesting data into Delphix for the purpose of provisioning dev and test instances first needs the creation of a data source in Delphix, called a "linked source" or "dSource" for short. The idea here is that a dSource is linked to a physical database whose data would be ingested into Delphix. The process of ingesting data into Delphix is initiated via the "SnapSync" workflow. When a SnapSync is initia ted, Delphix communicates with the database host and executes a backup of the database. This backup stream is the source of the data on Delphix for a given Oracle database. Subsequent SnapSync jobs capture changes that have happened since the last SnapSync. Delphix can also fetch archived logs as they are generated, allowing for point in time provisioning of a virtual database or V2P operation.

The current approach has several requirements which can make it unsuitable for some environments:

1. To be able to initiate a backup and fetch archived logs from the source environment, the Delphix Engine requires OS level connectivity and reasonably high levels of database privileges. In many cases this can pose security concerns.

2. In most cases, production databases will have existing backup procedures in place. The overhead of performing specific backups for the Delphix engine, and transferring the backups and archived logs to the engine can be a duplication of effort and poor use of resources.

To help customers use their existing backups (either RMAN or 3rd party based), Delphix, starting from 7.0.0.0, introduced the "Staging Push" feature.

Staging Push

Staging Push introduces the concept of a staging instance for Oracle dSource ingestion. Storage for the staging instance is provided by the Delphix Engine via NFS, Existing backups are then used to restore the source database onto the Delphix provided storage, at which point a dSource SnapSync can be initiated. Staging Push enhances the data ingestion mechanism by providing the following:

- » Any backup vendor and backup location can be used to restore the backups.
- » The staging database can also be synchronized with the source database using various database replication tools. Delphix has validated Data Guard and Active Data Guard for this process.
- » No access to the production environment or database is required at any point by Delphix, providing true “zero production touch” capability.
- » Unlike with the Delphix OBI plugin, Staging Push snapshots are provisioned in the exact same manner as regular Oracle snapshots, providing feature parity with traditional Oracle dSources.

Data Ingestion with Staging Push

Once a Staging Push dSource is created, a running Oracle instance will be available on the Oracle staging host for ingesting production data. Before initiating Snapsync for this dSource, the staging instance can be populated with production data in the following ways:

1. Restore existing database backups on the staging database. A pre-sync hook can be used to automate the restore process.
2. Setup the staging database as a physical standby database. A pre-sync and a post-sync hook can be used to stop/start managed recovery.

The staging database needs to be active only during dSource SnapSync operation. It can be shutdown after the snapshot to conserve system resources on the Oracle staging host. pre-sync and post-sync hooks can start/stop the staging database before/after taking snapshots.

A staging push dSource can be setup using following steps:

1. Create a dSource from UI, CLI or API.
2. Configure pre-sync and post sync hooks as per the decided ingestion strategy or set up external scripts to populate the staging database.
3. Take dSource snapshot manually or by configuring a SnapSync policy.

Link/Sync a dSource

Following are the steps to link and sync a multitenant database but similar operations can be performed for a non-multitenant database as well by selecting appropriate parameters in the Link dSource wizard.

1. Link a CDB dSource. Refer the Delphix documentation here to link the dSource.
2. Link a PDB dSource.
3. On the Oracle staging host, Delphix storage will be mounted for both CDB and PDB under the specified mount directory.
4. Populate the staging database with the latest data from the source database.

   Note: The datafiles in the staging database need to be placed in following directories:

   » For Non Multi Tenant databases: MOUNT_BASE/<Database_Unique_Name>/datafile

   » For CDBs and PDB$SEED: MOUNT_BASE/<CDB_Unique_Name>/datafile

   » For PDBs: MOUNT_BASE/<PDB_Name>-<CDB_Unique_Name>/datafile
5. The archive logs can be placed in the mounted 'archive' directory to enable point in time provisioning from the dSource snapshots.

   Note: The archive log files must be placed in following directories:

   » For Non Multi Tenant databases: MOUNT_BASE/<Database_Unique_Name>/datafile

   » For Non Multi Tenant databases: MOUNT_BASE/<Database_Unique_Name>/archive

   » For CDBs: MOUNT_BASE/<CDB_Unique_Name>/archive
   All the logs should be in the 'archive' directory only. Logs placed in subdirectories of the 'archive' directory are not supported as of writing this whitepaper.
6. Take a snapshot of the dSource. Refer to the Delphix documentation here to take the snapshot.

Configure Hooks

Refer the Delphix documentation here to configure hooks. Hooks can be configured either while creating the dSource or even after the dSource is created.

Data Ingestion from Database Backups

The Staging database can be updated with latest data from the backups of the source database by configuring the restore process as a presync hook.

Sample hook scripts to restore full and incremental backups are given below using the environment variables available in the hooks. Both the scripts can be configured as a pre-sync hook. The scripts should be modified as per the requirements before use.

Non Multi Tenant Database

Full (Level 0) Restore

```
# Statement of Support
# This software is provided as-is, without warranty of any kind or commercial support
# through Delphix. See the associated license for additional details. Questions,
# issues, feature requests, and contributions should be directed to the community as
# outlined in the Delphix Community Guidelines.

# Input Section
BACKUP_LOCATION=<backup-location>
CF_AUTOBACKUP_LOCATION=$BACKUP_LOCATION/%F
DB_ID=<db-id>
LOG_FILE=<hook-log-file-location>

# Declarations
DB_NAME=$DELPHIX_DATABASE_NAME
DB_MOUNT_PATH=${DELPHIX_MOUNT_PATH}/${DELPHIX_DATABASE_UNIQUE_NAME}/datafile

export ORACLE_SID=${ORACLE_SID}
export SQLPLUS_DML_MODIFIERS='set heading off pagesize 0 echo off define off feedback
off verify off trimout on timing off;'

# Exit if it is an incremental restore
[ -n "$(ls -A --ignore "*.ora" $DB_MOUNT_PATH)" ] && exit 0

# Restore the control file
rman target / <<-EOF >> $LOG_FILE
      run {
         set dbid $DB_ID;

         # Set autobackup format if the backup is at non-default location
         set controlfile autobackup format for device type disk to '$CF_
         AUTOBACKUP_LOCATION';

         restore CONTROLFILE from autobackup;
      }
      exit;
EOF

# Mount the database
sqlplus "/ as sysdba" <<-EOF >> $LOG_FILE
         alter database mount;
         exit;
EOF

rman target "/" <<-EOF >> $LOG_FILE
      run {
            set newname for database to new;

            # Catalog backup files
            catalog start with '$BACKUP_LOCATION' noprompt;

            # Restore database
            restore database;

            switch datafile all;
         }
         exit;
EOF

# Find the recovery SCN
RECOVERY_SCN=`sqlplus -S -R 3 "/ as sysdba" <<-EOF
      $SQLPLUS_DML_MODIFIERS
      SELECT nvl(MIN(latest_change), 0)
      FROM (
         SELECT max(next_change#) latest_change
         FROM v\\$backup_redolog brd, v\\$database db, v\\$backup_piece pc
         WHERE brd.resetlogs_change# = db.resetlogs_change#
           AND brd.resetlogs_time = db.resetlogs_time
           AND brd.set_stamp = pc.set_stamp
           AND brd.set_count = pc.set_count
           AND decode(media, null, 'DISK', 'TAPE') = 'DISK' GROUP BY thread#
      );
      exit;
EOF`

rman target "/" <<-EOF >> $LOG_FILE
        # Recover database
        run {
          set until scn $RECOVERY_SCN;
          recover database;
        }
        exit;
EOF
```

Incremental (Level-1) Restore

```
# Statement of Support
# This software is provided as-is, without warranty of any kind or commercial support
# through Delphix. See the associated license for additional details. Questions,
# issues, feature requests, and contributions should be directed to the community as
# outlined in the Delphix Community Guidelines.

# Input Section
BACKUP_LOCATION=<backup-location>
CF_AUTOBACKUP_LOCATION=$BACKUP_LOCATION/%F
```

```
DB_ID=<db-id>
LOG_FILE=<hook-log-file-location>

# Declarations
DB_NAME=$DELPHIX_DATABASE_NAME
DB_MOUNT_PATH=${DELPHIX_MOUNT_PATH}/${DELPHIX_DATABASE_UNIQUE_NAME}/datafile

export ORACLE_SID=${ORACLE_SID}
export SQLPLUS_DML_MODIFIERS='set heading off pagesize 0 echo off define off
feedback off verify off trimout on timing off;'

# Exit if not an incremental restore
[ -z "$(ls -A --ignore "*.ora" $DB_MOUNT_PATH)" ] && exit 0

# Restart the database in NOMOUNT mode
sqlplus "/ as sysdba" <<-EOF >> $LOG_FILE
      startup force nomount;
      exit;
EOF

# Restore the control file
rman target / <<-EOF >> $LOG_FILE
      run {
         set dbid $DB_ID;

         # Set autobackup format if the backup is at non-default location
         set controlfile autobackup format for device type disk to
         '$CF_AUTOBACKUP_LOCATION';

         restore controlfile from autobackup;
      }
      exit;
EOF

# Mount the database
sqlplus "/ as sysdba" <<-EOF >> $LOG_FILE
      alter database mount;
      exit;
EOF

rman target "/" <<-EOF >> $LOG_FILE
      # Catalog existing database files
      CATALOG START WITH '$DB_MOUNT_PATH' noprompt;

      # Switch datafiles to copy
      switch database to copy;

      # Catalog backup files
      catalog start with '$BACKUP_LOCATION' noprompt;

      exit;
EOF
```

```
# Find the recovery SCN
RECOVERY_SCN=`sqlplus -S -R 3 "/ as sysdba" <<-EOF
      $SQLPLUS_DML_MODIFIERS
      SELECT nvl(MIN(latest_change), 0)
      FROM (
         SELECT max(next_change#) latest_change
         FROM v\\$backup_redolog brd, v\\$database db, v\\$backup_piece pc
         WHERE brd.resetlogs_change# = db.resetlogs_change#
           AND brd.resetlogs_time = db.resetlogs_time
           AND brd.set_stamp = pc.set_stamp
           AND brd.set_count = pc.set_count
           AND decode(media, null, 'DISK', 'TAPE') = 'DISK' GROUP BY thread#
      );
      exit;
EOF`

rman target "/" <<-EOF >> $LOG_FILE
      # Recover database
      run {
         set until scn $RECOVERY_SCN;
         recover database;
      }
      exit;
EOF
```

Multi-Tenant Database

Full (Level 0) Restore

```
# Statement of Support
# This software is provided as-is, without warranty of any kind or commercial support
# through Delphix. See the associated license for additional details. Questions,
# issues, feature requests, and contributions should be directed to the community as
# outlined in the Delphix Community Guidelines.

# Input Section
BACKUP_LOCATION=<backup-location>
CF_AUTOBACKUP_LOCATION=$BACKUP_LOCATION/%F
DB_ID=<db-id>
BYSTANDER_PDB_CON_IDS=<con_ids of bystander PDBS, separated by comma, e.g. 4,5>
LOG_FILE=<hook-log-file-location>

# Declarations
PDB_MOUNT_PATH=${DELPHIX_MOUNT_PATH}/${DELPHIX_PDB_NAME}-${DELPHIX_DATABASE_UNIQUE_
NAME}/datafile
CDB_MOUNT_PATH=${DELPHIX_MOUNT_PATH}/${DELPHIX_DATABASE_UNIQUE_NAME}/datafile
```

```
# Exit if it is an incremental restore
[ -n "$(ls -A $PDB_MOUNT_PATH)" ] && exit 0

export ORACLE_SID=${ORACLE_SID}
export SQLPLUS_DML_MODIFIERS='set heading off pagesize 0 echo off define off feedback
off verify off trimout on timing off;'

# Restore the control file
rman target / <<-EOF >> $LOG_FILE
      run {
         set dbid $DB_ID;

         # Set autobackup format if the backup is at non-default location
         set controlfile autobackup format for device type disk to
         '$CF_AUTOBACKUP_LOCATION';

         restore CONTROLFILE from autobackup;

      }
      exit;
EOF

# Mount the database
sqlplus "/ as sysdba" <<-EOF >> $LOG_FILE
        alter database mount;
        exit;
EOF

# Restore database
rman target "/" <<-EOF >> $LOG_FILE
      run {
         # Catalog backup files
         CATALOG START WITH '$BACKUP_LOCATION' NOPROMPT;

         set newname for database to new;

         alter system set DB_CREATE_FILE_DEST='$CDB_MOUNT_PATH';

         # Restore CDB$ROOT
         restore force device type disk database root;

         # Restore PDB$SEED
         restore force device type disk pluggable database "pdb\$seed";

         # Restore PDB. Restore only that PDB for which the dSource is created.
         alter system set DB_CREATE_FILE_DEST='$PDB_MOUNT_PATH';

         restore force device type disk PLUGGABLE DATABASE ${DELPHIX_PDB_NAME};
         switch datafile all;
      }
      exit;
EOF
```

```
# Find the recovery SCN
RECOVERY_SCN=`sqlplus -S -R 3 “/ as sysdba” <<-EOF
      $SQLPLUS_DML_MODIFIERS
      SELECT nvl(MIN(latest_change), 0)
      FROM (
         SELECT max(next_change#) latest_change
         FROM v\\$backup_redolog brd, v\\$database db, v\\$backup_piece pc
         WHERE brd.resetlogs_change# = db.resetlogs_change#
           AND brd.resetlogs_time = db.resetlogs_time
           AND brd.set_stamp = pc.set_stamp
           AND brd.set_count = pc.set_count
           AND decode(media, null, ‘DISK’, ‘TAPE’) = ‘DISK’ GROUP BY thread#
      );
      exit;
EOF`
# Get a list of all the tablespaces of bystander PDBs to skip them during recovery
# This generates a string like “CDOMLOSR1E0FPDB2:SYSTEM,CDOMLOSR1E0FPDB2:SYSAUX”
BYSTANDER_PDB_TS=`sqlplus -S -R 3 “/ as sysdba” <<-EOF
         $SQLPLUS_DML_MODIFIERS
      set linesize 4096 wrap on recsep off tab off
      set serveroutput on
      DECLARE
         v_ts varchar(5000);
      BEGIN
         FOR pdb_ts IN (SELECT p.name  || ‘:’ || ts.name pdb_ts_name FROM
         v\\$tablespace ts, v\\$pdbs p WHERE p.con_id = ts.con_id and p.con_id in
         ($BYSTANDER_PDB_CON_IDS))
         LOOP
           v_ts := v_ts || pdb_ts.pdb_ts_name || ‘,’;
         END LOOP;
         DBMS_OUTPUT.PUT_LINE(TRIM(TRAILING ‘,’ FROM v_ts));
      END;
      /
      exit;
EOF`
# Recover database
rman target “/” <<-EOF >> $LOG_FILE
      run {
         set until scn $RECOVERY_SCN;
         recover database skip forever tablespace $BYSTANDER_PDB_TS;
      }
      exit;
EOF
```

Incremental (Level 1) Restores

```
# Statement of Support
# This software is provided as-is, without warranty of any kind or commercial support
# through Delphix. See the associated license for additional details. Questions,
# issues, feature requests, and contributions should be directed to the community as
# outlined in the Delphix Community Guidelines.

# Input Section
BACKUP_LOCATION=<backup-location>
CF_AUTOBACKUP_LOCATION=$BACKUP_LOCATION/%F
DB_ID=<db-id>
BYSTANDER_PDB_CON_IDS=<con_ids of bystander PDBS, separated by comma, e.g. 4,5>
LOG_FILE=<hook-log-file-location>

# Declarations
PDB_MOUNT_PATH=${DELPHIX_MOUNT_PATH}/${DELPHIX_PDB_NAME}-${DELPHIX_DATABASE_UNIQUE_
NAME}/datafile
CDB_MOUNT_PATH=${DELPHIX_MOUNT_PATH}/${DELPHIX_DATABASE_UNIQUE_NAME}/datafile

# Exit if it is not an incremental restore
[ -z “$(ls -A $PDB_MOUNT_PATH)” ] && exit 0

export ORACLE_SID=${ORACLE_SID}
export SQLPLUS_DML_MODIFIERS=’set heading off pagesize 0 echo off define off feedback
off verify off trimout on timing off;’

# Restart the database in NOMOUNT mode
sqlplus “/ as sysdba” <<-EOF >> $LOG_FILE
      startup force nomount;
      exit;
EOF

# Restore the control file
rman target / <<-EOF >> $LOG_FILE
      run {
         set dbid $DB_ID;

         # Set autobackup format if the backup is at non-default location
         set controlfile autobackup format for device type disk to ‘$CF_
AUTOBACKUP_LOCATION’;

         restore CONTROLFILE from autobackup;
      }
      exit;
EOF

# Mount the database
sqlplus “/ as sysdba” <<-EOF >> $LOG_FILE
       alter database mount;
       exit;
EOF
```

```
# Switch to existing datafiles
rman target “/” <<-EOF >> $LOG_FILE
      # Catalog CDB datafiles
      CATALOG START WITH ‘${DELPHIX_MOUNT_PATH}/${DELPHIX_DATABASE_UNIQUE_NAME}’
      NOPROMPT;

      # Catalog PDB datafiles
      CATALOG START WITH ‘${DELPHIX_MOUNT_PATH}/${DELPHIX_PDB_NAME}-${DELPHIX_
      DATABASE_UNIQUE_NAME}’ NOPROMPT;

      # Switch to existing datafiles
      switch database root to copy;

      switch pluggable database ‘pdb\$seed’ to copy;

      switch pluggable database ${DELPHIX_PDB_NAME} to copy;

      # Catalog backup files
      CATALOG START WITH ‘$BACKUP_LOCATION’ NOPROMPT;

      exit;
EOF

# Find the recovery SCN
RECOVERY_SCN=`sqlplus -S -R 3 “/ as sysdba” <<-EOF
      $SQLPLUS_DML_MODIFIERS
      SELECT nvl(MIN(latest_change), 0)
      FROM (
         SELECT max(next_change#) latest_change
         FROM v\\$backup_redolog brd, v\\$database db, v\\$backup_piece pc
         WHERE brd.resetlogs_change# = db.resetlogs_change#
           AND brd.resetlogs_time = db.resetlogs_time
           AND brd.set_stamp = pc.set_stamp
           AND brd.set_count = pc.set_count
           AND decode(media, null, ‘DISK’, ‘TAPE’) = ‘DISK’ GROUP BY thread#
      );
      exit;
EOF`

# Get a list of all the tablespaces of bystander PDBs to skip them during recovery
# This generates a string like “CDOMLOSR1E0FPDB2:SYSTEM,CDOMLOSR1E0FPDB2:SYSAUX”
BYSTANDER_PDB_TS=`sqlplus -S -R 3 “/ as sysdba” <<-EOF
      $SQLPLUS_DML_MODIFIERS
      set linesize 4096 wrap on recsep off tab off
      set serveroutput on
      DECLARE
           v_ts varchar(5000);
      BEGIN
           FOR pdb_ts IN (SELECT p.name  || ‘:’ || ts.name pdb_ts_name FROM
           v\\$tablespace ts, v\\$pdbs p WHERE p.con_id = ts.con_id and p.con_id in
           ($BYSTANDER_PDB_CON_IDS))
```

```
                LOOP
                        v_ts := v_ts || pdb_ts.pdb_ts_name || ‘,’;
                END LOOP;
                DBMS_OUTPUT.PUT_LINE(TRIM(TRAILING ‘,’ FROM v_ts));
            END;
            /
            exit;
        EOF`

        # Recover database
        rman target “/” <<-EOF >> $LOG_FILE
            run {
             set until scn $RECOVERY_SCN;
             recover database skip forever tablespace $BYSTANDER_PDB_TS;
            }
            exit;
        EOF
```

Data Ingestion from Staging Database Setup as Physical Standby

The staging database can be setup as a physical standby using the following steps:

1. Link a CDB dSource. Refer the Delphix documentation here to link the dSource.
   The dSource should be created with the following additional settings:
    a. Select the “physical standby” option.
    b. Set additional config params such as LOG_ARCHIVE_CONFIG on the Staging DB (needed for log apply between Primary and Standby)
       `LOG_ARCHIVE_CONFIG=”’DG_CONFIG=(PRIMARY_DB_UNIQUE_NAME, STANDBY_DB_UNIQUE_NAME)’”`

2. Link the PDB dSource.

3. Set up Staging Database as Standby under Oracle Data Guard Configuration.
    » Create a STANDBY control file for the Staging Database
      - Create the STANDBY control file
        ```
        ALTER DATABASE CREATE STANDBY CONTROLFILE AS
        ‘/path/to/standby.ctl’ REUSE;
        ```
      - Replace the existing control file (which was restored from the backup) with the STANDBY control file
        ```
        mv /path/to/standby.ctl /{DELPHIX_MOUNT_PATH}/${DELPHIX_DATABASE_UNIQUE_NAME}/
        datafile/control.ctl
        ```
      - Restart the Staging Database
        ```
        STARTUP FORCE MOUNT
        pfile=’/{DELPHIX_MOUNT_PATH}/${DELPHIX_DATABASE_UNIQUE_NAME}/datafile/init.
        ora’;
        ```

» Copy Primary DB's password file to the Staging DB on the Staging host

```
scp -r $ORACLE_HOME/dbs/orapw<PRIMARY_DB_SID> oracle@<STAGING_HOST>:$ORACLE_HOME/
dbs/orapw<STANDBY_DB_SID>
```

» Create Standby Redo Logs (SRL) groups on both PRIMARY and STANDBY

```
# No. of SRL groups must be one more than the no. of online redo groups on Primary
and their size should be the same as that of the primary's redo group size.

# On Primary
ALTER DATABASE ADD STANDBY LOGFILE GROUP 4 SIZE 50M;
ALTER DATABASE ADD STANDBY LOGFILE GROUP 5 SIZE 50M;
ALTER DATABASE ADD STANDBY LOGFILE GROUP 6 SIZE 50M;
ALTER DATABASE ADD STANDBY LOGFILE GROUP 7 SIZE 50M;

# On Standby
ALTER DATABASE ADD STANDBY LOGFILE GROUP 4 SIZE 50M;
ALTER DATABASE ADD STANDBY LOGFILE GROUP 5 SIZE 50M;
ALTER DATABASE ADD STANDBY LOGFILE GROUP 6 SIZE 50M;
ALTER DATABASE ADD STANDBY LOGFILE GROUP 7 SIZE 50M;
```

» Start Managed Recovery Process (MRP) on STANDBY

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION USING
CURRENT LOGFILE;
```

» Set Parameters on the PRIMARY once the STANDBY is mounted

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(PRIMARY_DB_UNIQUE_NAME, STANDBY_DB_
UNIQUE_NAME)';

ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE="(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)
(HOST=<STANDBY_HOST>)(PORT=1521))(CONNECT_DATA=(SID=<STANDBY_DB_SID>)
(SERVER=DEDICATED)))" LGWR SYNC VALID_FOR=(ONLINE_LOGFILE,PRIMARY_ROLE) DB_UNIQUE_
NAME=<STANDBY_DB_UNIQUE_NAME>' SCOPE=BOTH;

ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2='ENABLE' SCOPE=BOTH;

ALTER SYSTEM SET STANDBY_FILE_MANAGEMENT='AUTO' SCOPE=BOTH;
```

4. Take a snapshot of the dSource.
   Following scripts can be configured as pre-sync and post-sync hooks to start and stop the managed recovery process. The script should be modified as per the requirements before use.

   Pre-sync hook:

```
   # Statement of Support
   # This software is provided as-is, without warranty of any kind or commercial
support
   # through Delphix. See the associated license for additional details. Questions,
   # issues, feature requests, and contributions should be directed to the community as
   # outlined in the Delphix Community Guidelines.
```

```
  sqlplus “/ as sysdba” <<-EOF >> ~/pre-hook.log
       alter database recover managed standby database cancel;
       shutdown immediate;
       startup mount
        pfile='/{DELPHIX_MOUNT_PATH}/${DELPHIX_DATABASE_UNIQUE_NAME}/datafile/init.ora'
       ;
       exit;
EOF
```

Post-sync hook:

```
# Statement of Support
# This software is provided as-is, without warranty of any kind or commercial support
# through Delphix. See the associated license for additional details. Questions,
# issues, feature requests, and contributions should be directed to the community as
# outlined in the Delphix Community Guidelines.

sqlplus \”/ as sysdba\” <<-EOF >> ~/post-hook.log
       alter database open read only;
       alter database recover managed standby database disconnect from session using
       current logfile;
       exit;
EOF
```

Partial Ingestion with Staging Push

During the restore process, only selected tablespaces can be restored on the staging database. The restore and recovery statements can be used as follows to skip selected tablespaces.

```
# Statement of Support
# This software is provided as-is, without warranty of any kind or commercial support
# through Delphix. See the associated license for additional details. Questions,
# issues, feature requests, and contributions should be directed to the community as
# outlined in the Delphix Community Guidelines.
run {
       restore database skip forever tablespace <tablespace-list>;
}

run {
       set until scn $RECOVERY_SCN;
       recover database skip forever tablespace <tablespace-list>;
}
```

Logsync for Point-In-Time Provisioning

Archived logs restored on the mounted 'archive' directory on the staging host during the restore process enable point-in-time provisioning from dSource snapshots.

The following pre-sync hook script can be used to automatically restore archive logs from the latest backup to the mounted 'archive' directory:

```
# Statement of Support
# This software is provided as-is, without warranty of any kind or commercial support
# through Delphix. See the associated license for additional details. Questions,
# issues, feature requests, and contributions should be directed to the community as
# outlined in the Delphix Community Guidelines.

# Input Section
BACKUP_LOCATION=<backup-location>
LOG_FILE=<hook-log-file-location>

# Declarations
ARCHIVE_MOUNT_PATH=${DELPHIX_MOUNT_PATH}/${DELPHIX_DATABASE_UNIQUE_NAME}/archive

export ORACLE_SID=${ORACLE_SID}
export SQLPLUS_DML_MODIFIERS='set heading off pagesize 0 echo off define off feedback
off verify off trimout on timing off;'

rman target / <<-EOF >> $LOG_FILE
run {
        # Uncatalog all the existing backups. 'noprompt' clause may not work in
        # Oracle version 12.1.0.1.
        change backup uncatalog noprompt;

        # Uncatalog all the existing archive log entries
        change archivelog all uncatalog;

        # Catalog backup files
        catalog start with '$BACKUP_LOCATION' noprompt;
}
exit;
EOF

# Find the lowest sequence number to restore
MIN_SEQ=`sqlplus -S -R 3 "/ as sysdba" <<-EOF
        $SQLPLUS_DML_MODIFIERS
        select min(sequence#)
        from v\\$backup_redolog vbr, v\\$backup_piece vbp, v\\$database_incarnation vdi
        where vbr.resetlogs_change# = vdi.resetlogs_change#
        and vbr.set_stamp = vbp.set_stamp
        and vdi.status = ‘CURRENT’
        and vbp.status = ‘A’;
        exit;
EOF`

# Find the highest sequence number to restore
MAX_SEQ=`sqlplus -S -R 3 “/ as sysdba” <<-EOF
        $SQLPLUS_DML_MODIFIERS
        select max(sequence#)
        from v\\$backup_redolog vbr, v\\$backup_piece vbp, v\\$database_incarnation vdi
        where vbr.resetlogs_change# = vdi.resetlogs_change#
        and vbr.set_stamp = vbp.set_stamp
        and vdi.status = ‘CURRENT’
        and vbp.status = ‘A’;
        exit;
EOF`
# Exit if the min or max seq values are not found
if [[ -z $MIN_SEQ ]]; then
        echo “No logs to restore” >> $LOG_FILE
        exit;
fi
if [[ -z $MAX_SEQ ]]; then
        echo “No logs to restore” >> $LOG_FILE
        exit;
fi
# Set the archive log file location
sqlplus “/ as sysdba” <<-EOF >> $LOG_FILE
        alter system set log_archive_dest_1=’location=$ARCHIVE_MOUNT_PATH MANDATORY’;
        exit;
EOF
# Restore the archive log files from backup
rman target / <<-EOF >> $LOG_FILE
        run {
                restore force device type disk archivelog from logseq $MIN_SEQ until
                logseq $MAX_SEQ;
        }

        exit;
EOF
```

Configure TDE for Staging Databases

Refer Oracle documentation to setup TDE for a staging database in different configurations. As an example, following steps can be used to configure TDE for a staging database associated with a Staging Push dSource.

Please note that in case of multitenant databases, a Staging Push CDB cannot have both encrypted and non-encrypted PDBs.

```
# Statement of Support
# This software is provided as-is, without warranty of any kind or commercial support
# through Delphix. See the associated license for additional details. Questions,
# issues, feature requests, and contributions should be directed to the community as
# outlined in the Delphix Community Guidelines.
```

1. Link a non-multitenant/CDB dSource. Refer the Delphix documentation here to link the dSource. On successful linking, a new staging Oracle instance will be created on the remote Oracle host.
2. In the staging Oracle installation, configure TDE keystore location in “sqlnet.ora”:

   ```
   ENCRYPTION_WALLET_LOCATION=
           (SOURCE=
                   (METHOD=FILE)
                   (METHOD_DATA=
                   (DIRECTORY=<staging keystore location>)))
   ```

   For Oracle 18+ versions, the keystore location can also be configured by setting the “wallet_root” system parameter:

   ```
   ALTER SYSTEM SET WALLET_ROOT='<staging keystore location>)' SCOPE=BOTH SID='*';
   ALTER SYSTEM SET TDE_CONFIGURATION="KEYSTORE_CONFIGURATION=<keystore_type>"
   SCOPE=BOTH SID='*';
   ```

3. Create the keystore directory:

   ```
   mkdir -p <staging keystore location>
   ```

4. Create an empty keystore at the keystore location:

   ```
   SQL> administer key management create keystore ‘<staging keystore location>’
   identified by “<staging keystore password>”;
   keystore altered.
   ```

5. Copy the source database keystore to the host where staging Oracle instance resides. Merge the source keystore into the newly created empty keystore for staging database.

   ```
   SQL> administer key management merge keystore ‘<source keystore location>’
   identified by “<source keystore password>” into existing keystore ‘<staging
   keystore location>’ identified by “<staging keystore password>” with backup;

   keystore altered.
   ```

Note that this step must be executed for every PDB coming from a different source CDB.

6. Verify the status of the staging keystore:

```
SQL> select * from v$encryption_wallet;

WRL_TYPE     WRL_PARAMETER            STATUS

FILE         /u01/tde/CDOMLOSRE17C/   CLOSED
```

7. Open the staging keystore:

```
SQL> administer key management set keystore open force keystore identified by
"<staging keystore password>";
keystore altered.
```

8. Verify the status of the staging keystore:

```
SQL> select * from v$encryption_wallet;

WRL_TYPE     WRL_PARAMETER            STATUS

FILE         /u01/tde/CDOMLOSRE17C/   OPEN
```

After this setup, an encrypted source database backup can be restored and recovered on the staging instance before taking a dSource snapshot. The staging database can also be setup as a physical standby database to automatically receive incremental data from the source database.

For provisioning, standard Delphix procedures to provision a non-mutitenant database or a virtual pluggable database with TDE can be used as is.

Conclusion

Staging push provides a flexible approach to ingestion and lets the user control how data is ingested into Delphix with zero production touch. Staging Push dSources don't need any access to the source database at any point during the ingestion process. Also since the database is in consistent state while taking a snapshot, provisioning from these snapshots is faster as no database recovery is needed during provisioning.

Ingesting Oracle Backups Into Delphix

Delphix

Delphix is the industry leader for DevOps test data management.

Businesses need to transform application delivery but struggle to balance speed with data security and compliance. Our DevOps Data Platform automates data security, while rapidly deploying test data to accelerate application releases. With Delphix, customers modernize applications, adopt multicloud, achieve CI/CD, and recover from downtime events such as ransomware up to 2x faster.

Leading companies, including Choice Hotels, Banco Carrefour, and Fannie Mae, use Delphix to accelerate digital transformation and enable zero trust data management. Visit us at www.delphix.com.

449