Oracle GoldenGate 12c Tutorial

Oracle to Oracle Replication with Oracle Multitenant

Version 12.1

Document ID 1960719.1

Tracy West
Consulting Solution Architect
Fusion Middleware Architects Team: The A-Team

Disclaimer

This sample code is provided for educational purposes only and not supported by Oracle Support Services. It has been tested internally, however, and works as documented. We do not guarantee that it will work for you, so be sure to test it in your environment before relying on it.

Proofread this sample code before using it! Due to the differences in the way text editors, e-mail packages and operating systems handle text formatting (spaces, tabs and carriage returns), this sample code may not be in an executable state when you first receive it. Check over the sample code to ensure that errors of this type are corrected.

This document touches briefly on many important and complex concepts and does not provide a detailed explanation of any one topic since the intent is to present the material in the most expedient manner. The goal is simply to help the reader become familiar enough with the product to successfully design and implement an Oracle GoldenGate environment. To that end, it is important to note that the activities of design, unit testing and integration testing which are crucial to a successful implementation have been intentionally left out of the guide. All the sample scripts are provided as is. Oracle consulting service is highly recommended for any customized implementation.

Table of Contents

Introduction

Oracle GoldenGate Extract, Replicat and associated utilities enable you to create, load and refresh one Oracle database to another Oracle database.

This tutorial provides a quick overview of Oracle to Oracle database replication using Integrated Extract and Integrated Replicat for version 12c and above. Extraction from a Multitenant database is only supported by Integrate Extract. For more detailed information, please consult the Oracle GoldenGate Administration Guide.

This tutorial may be read to get a general overview of how Extract and Replicat operate. Alternatively, you can follow along each step of the way. This tutorial utilizes an extract pulling data from a single PDB and sending the data across the network to the target system. Once the data is written to trail files on the target system it will be applied to a single PDB by a replicat. In a production environment, an OGG Extract pump would be utilized to minimize the risk of a replication outage due to a network issue.

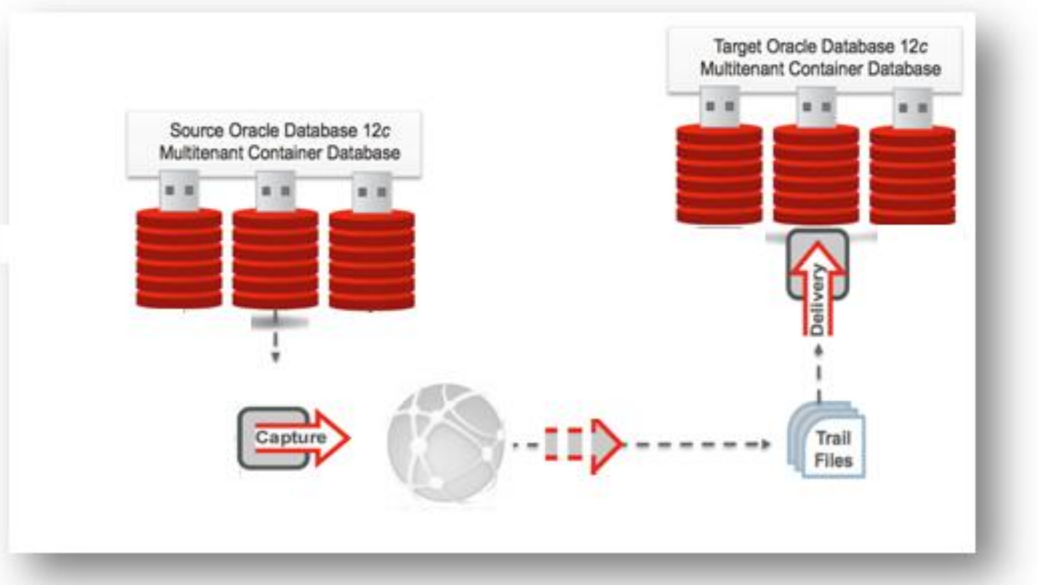
Target Oracle Database 12c
Multitenant Container Database
Source Oracle Database 12c
Multitenant Container Database
Delivery
Capture
Trail
Files

Figure 1 – Tutorial Architecture

Prerequisites

If you plan to execute the instructions in the tutorial, make sure all software is already installed. The reader should also be familiar with basic Multitenant architecture and functionality. Multitenant is an Oracle 12c RDBMs feature. It is assumed for this tutorial that both the source and target are running Oracle 12c RDBMS. For Oracle 12.1.0.2 and above, the init.ora parameter, `enable_goldengate_replication` must be set to `TRUE` in both the source and target databases. This parameter must be set in the CDB$ROOT. Also the PDBs being replicated need to be accessible and should have entries in the tnsnames.ora. These PDBs should also be open and available.

The following table describes items that are referred to throughout the tutorial. You will need to identify your installation-specific values and substitute them as you go.

Item	Tutorial Reference	Description
Unix Programs	/ggs	Directory of Unix GoldenGate installation.
Unix Parameter Files	/ggs/dirprm	Directory for GoldenGate parameter files.
Unix Report Files	/ggs/dirrpt	Directory for output from GoldenGate programs.
Unix Definitions Files	/ggs/dirdef	Directory for generated Oracle DDL and definition files.
GGS temporary storage	/ggs/dirdat	Directory to hold temporary Extract trails
Oracle Logon	userid, password	User ID and password for the source or target database. When implementing Integrated Extract or Replicat, this user must be granted admin privileges with the ***DBMS_GOLDENGATE_AUTH procedure*** on the source database.
Unix System Network Address	unixserver2 unixserver1	IP address/hostname of the target Unix system in network. IP address/hostname of the source Unix system in network

The source Oracle database tables used in this tutorial can be created and loaded with sample data using the following scripts from the OGG installation directory. Make sure to login directly to the PDB with the schema owner.

Tutorial pre-setup:

```
PDB=>pdb1
PDB Schema =>test
PDB Schema Password => test_pass
PDB service name => 'PDB1'
```

Create Source Tables and Load with Data using scripts from the OGG installation directory.

```
%(unixserver1) sqlplus test/test_pass@pdb1
SQLPLUS> @demo_ora_create
SQLPLUS> @demo_ora_insert
SQLPLUS> exit
```

Create Target Oracle Tables used in this tutorial using the following script from the OGG installation directory.

```
%(unixserver2) sqlplus test/test_pass@pdb1
SQLPLUS> @demo_ora_create
SQLPLUS> exit
```

Overview of Tutorial Tasks

This section outlines the steps required in each phase of database load and replication.

Extract and Replicat work together to keep the databases in sync near real-time via incremental transaction replication. Perform this function by

» Starting the Manager program on both the source and target systems.

» Adding supplemental transaction log data for update operations.

» Running the real-time Extract to retrieve and store the incremental changed data from the Oracle tables into trail files on the target Unix system.

Once the target database is created, it can be loaded with data from the Oracle source database. To load the target database via OGG tools by

» Running the initial load Extract to retrieve, convert and output data from the Oracle tables.

» Running the initial load Replicat to insert the initial data into the target database.

After initial synchronization,

» Start the real-time Replicat to replicate extracted data.

Once Extract and Replicat are running, changes are replicated perpetually. At this point, we will also demonstrate the following functions.

» How to retrieve information on Extract and Replicat status.

» How to gracefully stop replication.

» How to restart replication with transaction integrity.

Notes on Command Syntax

Commands throughout the tutorial make specific references to directories, file names, checkpoint group names, begin times, etc. Unless otherwise noted, these items do not have to correspond exactly in your environment; they are used to illustrated concrete examples. Where the prompt is written GGSCI {unixserver1}> the command should be executed on the source system. {unixserver2} indicates the target system.

For exact syntax, consult the Oracle GoldenGate Reference Guide.

Additional Requirements for Multitenant Container Databases

The following are the special requirements that apply to replication to and from multitenant container databases.

- All of the pluggable databases in the multitenant container database must have the same attributes, such as character set, locale, and case-sensitivity. The character set of the pluggable databases must be the same character set of the CDB or a binary subset of the CDB’s character set.
- Extract must operate in integrated capture mode.
- Extract must connect to the root container (cdb$root) as a common user in order to interact with the logmining server. To specify the root container, use the appropriate SQL*Net connect string for the database user that you specify with the USERID or USERIDALIAS parameter. For example: C##GGADMIN@FINANCE. For more information on ‘common user’ please refer to Oracle Database 12.1 documentation.
- A single Extract supports pulling data from multiple PDBs, but a single Replicat will only support applying data to 1 PDB. So if pulling data from multiple PDBs, multiple replicats will need to be created to apply all the data.
- The dbms_goldengate_auth.grant_admin_privilege package grants the appropriate privileges for capture and apply within a multitenant container databases. This includes the container parameter, which must be set to ALL for the extract user, as shown in the following example:

```
exec dbms_goldengate_auth.grant_admin_privilege('C##GGADMIN',container=>'all')
```

Collecting Information about your Multitenant Container Databases

The following are examples of how to determine information about your environment.

```
% sqlplus / as sysdba
```

The following Query should return ‘YES’ for a container database.

```
SQL> select cdb from v$database;
CDB
---
YES
```

The following Query will list all the PDBs.

```
SQL> select name, open_mode from v$pdbs;
NAME                 OPEN_MODE
---------------      ----------
PDB$SEED             READ ONLY <=== this is a seed PDB
PDBORCL              MOUNTED <===== this PDB at mount status
PDB1                 READ WRITE <== this is a PDB at read/write open status
```

The following command will open a specific PDB.

```
SQL> show con_name
CON_NAME
------------------------------
CDB$ROOT

SQL> alter session set container=PDBORCL;
Session altered.

SQL> show con_name <==== Current session container.
CON_NAME
------------------------------
```

```
PDBORCL

SQL> alter database PDBORCL open;
Database altered.

SQL> select name, open_mode from v$pdbs;
NAME                        OPEN_MODE
--------------------------------------
PDBORCL                     READ WRITE
```

The following query will list all the PDBs (including CDB) on services

From CDB root

```
SQL> connect /as sysdba
Connected.

SQL> select NAME,CON_ID,PDB from v$services;
NAME                        CON_ID  PDB
----------------------------  ------  ------------------------------
pdb1                        4       PDB1
pdborcl                     3       PDBORCL
orclXDB                     1       CDB$ROOT
orcl                        1       CDB$ROOT
SYS$BACKGROUND              1       CDB$ROOT
SYS$USERS                   1       CDB$ROOT
```

Note: the read only PDB$SEED is not listed here. Please refer to Oracle Database 12.1 documentation for more information on PDB$SEED.

This SQL issued from within a PDB, it will only list data related to that PDB

```
SQL> alter session set container=PDBORCL;
Session altered.

SQL> select NAME,CON_ID,PDB from v$services;
NAME CON_ID PDB
----------------------------  ------ ------------------------------
pdborcl                     3      PDBORCL
```

To connect to a PDB directly without 'alter session set container'. You must connect through the PDB network service.

```
$ sqlplus pdb1_u1/pdb1_u1@pdb1
SQL*Plus: Release 12.1.0.1.0 Production on Thu Dec 19 12:03:14 2013
Copyright (c) 1982, 2013, Oracle. All rights reserved.
Last Successful login time: Thu Dec 19 2013 12:02:49 -08:00
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing
options

SQL> show con_name
CON_NAME
------------------------------
PDB1
```

Prepare the Database for Replication and Start Capturing Changes

Before the initial load is started, supplemental logging needs to be enabled and real-time extract started. All changes occurring against source tables are automatically detected by Extract, then formatted and transferred near real-time to temporary files on the Unix file system. After initial load is completed, the data is read from these files and replicated to the target database by the Replicat.

Perform the following tasks to implement extraction and replication.

On the source system

- Create an OGG User
- Add supplemental log data for update operations.
- Create an Extract parameter file on source Unix system.
- Register Extract in database
- Set up an initial Extract checkpoint on source Unix system.
- Create Remote Trail File
- Start Extract

On the target system

- Create an OGG User
- Create a Replicat parameter file on target Unix system.
- Set up an initial Replicat checkpoint on target Unix system.

Create OGG Extract User

The Extract user for a Multitenant environment must be a common user and must log into the root container. In the following example, the extract userid is *c##ggadmin* using the password *ggadmin*. The following commands are executed in CDB$ROOT.

```
$(unixserver1)  sqlplus / as sysdba

SQL> create user C##GGADMIN identified by ggadmin;
User created.
SQL> exec
dbms_goldengate_auth.grant_admin_privilege('C##GGADMIN',container=>'ALL');
PL/SQL procedure successfully completed.
SQL> grant dba to c##ggadmin container=all;
Grant succeeded.
Please note the password may be case sensitive
(check sec_case_sensitive_logon):

SQL> connect C##GGADMIN/GGADMIN
ERROR:
ORA-01017: invalid username/password; logon denied
SQL> connect C##GGADMIN/ggadmin
```

```
Connected.
SQL> connect c##ggadmin/ggadmin
Connected.
```

The parameter in extract for userid and password should also be in the correct case.

```
e.g., userid c##ggadmin, password ggadmin
```

Adding supplemental log data

By default, Oracle only logs changed columns for update operations. Normally, this means that primary key columns are not logged during an update operation. However, Replicat requires the primary key columns in order to apply the update on the target system. The ADD TRANDATA or ADD SCHEMATRANDATA commands in GGSCI are used to cause Oracle to log primary key columns for all updates at the table. Also minimal supplemental logging needs to be enabled at the database level.

To enable minimal supplemental logging at the PDB level, issue the following command on the source Unix system.

```
$(unixserver1)  sqlplus / as sysdba
SQL> alter session set container=pdb1 ;
SQL> ALTER PLUGGABLE DATABASE ADD SUPPLEMENTAL LOG DATA;
Pluggable Database altered.
```

To add supplemental log data at the table level, issue the following commands on the source Unix system. For DDL replication which is enabled in this tutorial, use `ADD SCHEMATRANDATA` to ensure supplemental logging is automatically enabled for altered or new tables. `ADD TRANDATA` is demonstrated as an option for individual tables that you only want to replicate DML for.

```
$(unixserver1) cd /ggs
$(unixserver1) ggsci
GGSCI (unixserver1) > DBLOGIN USERID test@pdb1, PASSWORD test_pass
Successfully logged into database PDB1.

GGSCI (unixserver1) > ADD SCHEMATRANDATA pdb1.test
2014-10-17 13:23:47  INFO    OGG-01788  SCHEMATRANDATA has been added on
schematest.

2014-10-17 13:23:47  INFO    OGG-01976  SCHEMATRANDATA for scheduling columns
hs been added on schema test.
```

Or

```
GGSCI (unixserver1) 2> DBLOGIN USERID test@pdb1, PASSWORD test_pass
Successfully logged into database PDB1.

GGSCI (unixserver1) 3> ADD TRANDATA pdb1.test.tcustmer
Logging of supplemental redo data enabled for table PDB1.TEST.TCUSTMER.
TRANDATA for scheduling columns has been added on table 'PDB1.TEST.TCUSTMER'.

GGSCI (unixserver1) 4> ADD TRANDATA pdb1.test.tcustord
Logging of supplemental redo data enabled for table PDB1.TEST.TCUSTORD.
TRANDATA for scheduling columns has been added on table 'PDB1.TEST.TCUSTORD'.
```

The DBLOGIN command establishes a database connection for the specified user. The user is prompted for a password.

The ADD TRANDATA or ADD SCHEMATRANDATA command causes Oracle to log primary key columns for all update operations on the specified table. For Multitenant, the schema and PDB must be specified. ADD SCHEMATRANDATA should be used when DDL replication is enabled.

Prepare the Environment to Map and Collect Data

Prior to executing data extraction, which moves data from Oracle to Oracle in the Unix environment, perform the following task.

- Add the MGRPORT parameter to the Manager Parameter file. This ensures that server collector processes can be dynamically created on the remote system to receive and log data created by Extract.

Preparing Manager to Start Dynamic Server Collectors

The Server Collector program receives incoming data over a TCP/IP connection from Extract on the source Unix system, and then outputs the incremental changes to temporary storage on the target Unix system. The Server Collector is automatically started by the Manager process at the request of the Extract program whenever moving data between systems.

The Manager program provides a number of important functions, including monitoring critical system components and starting GoldenGate processes. Before running any other GoldenGate programs on Unix, you must start Manager.

Before starting Manager, you must edit Manager's startup parameter file (called /ggs/dirprm/mgr.prm) and add the PORT parameter. You can do this manually with a Unix editor, or you can use GGSCI to start the **vi** program for you with these commands:

```
GGSCI (unixserver1) > EDIT PARAMS MGR
```

In either case, add the following text to the MGR.PRM file, then save the file and quit.

```
PORT 7809
```

Manager is started via GGSCI using the following command.

```
GGSCI (unixserver1) > START MANAGER
```

If your target Oracle database is on another system, repeat the above steps on the target system. Start the target Manager process with the START MANAGER command.

```
GGSCI (unixserver2) > EDIT PARAMS MGR

PORT 7809

GGSCI (unixserver2) > START MANAGER
```

Configuring the Extract Parameter File

Most Extract parameters are entered into a parameter file. You may create your parameter file manually using **vi** or another editor on Unix. The parameter file name is the same name as the name of the group, in this case EXTORA. Issue the following command on the source system to launch **vi** from GGSCI :

```
GGSCI (unixserver1) > EDIT PARAMS EXTORA
```

You can also edit the file /ggs/dirprm/extora.prm directly from any other text editor.

Enter the following parameters into EXTORA. Note that the ordering of the parameters is important.

```
--
-- Extract parameter file to capture TCUSTORD
-- and TCUSTMER changes
--
EXTRACT EXTORA

USERID c##ggadmin, PASSWORD ggadmin

RMTHOST unixserver2, MGRPORT 7809

RMTTRAIL ./dirdat/rt

DDL INCLUDE MAPPED
DDLOPTIONS REPORT

LOGALLSUPCOLS
UPDATERECORDFORMAT COMPACT

TABLE pdb1.test.TCUSTMER;
TABLE pdb1.test.TCUSTORD;
```

Parameters explained:

EXTRACT EXTORA identifies the particular extract checkpoint group with which this parameter file is associated.

USERID and PASSWORD must match an existing account in the Oracle database. The active Oracle database is indicated by the user's ORACLE_SID environment variable. For Integrated Extract this user must have been granted Admin Privileges by using the 'DBMS_GOLDENGATE_AUTH' package executed by a SYS account. In this example the password is in clear text. The best practice is to encrypt the password.

RMTHOST identifies the system to which to output extracted database changes and must be specified before RMTTRAIL. MGRPORT specifies the port number on which Manager has been configured to listen for requests for Server Collectors. RMTHOST can also be specified as a standard TCP/IP host name.

RMTTRAIL specifies the file set to which database changes will be output. Changes detected on any TABLE specified below this entry will be output to the remote trail.

DDL INCLUDE MAPPED enables capture of DDL changes for mapped objects. DDL replication for Oracle 12c does not require any setup. It just needs to be enabled in the extract parameter file.

DDLOPTIONS REPORT enables DDL processing information to be written to the report file.

LOGALLSUPCOLS ensures the capture of the supplementally logged columns in the before image. This parameter is valid for any source database that is supported by Oracle GoldenGate. The database must be configured to log the before and after values of the primary key, unique indexes, and foreign keys.

UPDATERECORDFORMAT parameter set to COMPACT causes Extract to combine the before and after images of an UPDATE operation into a single record in the trail. This parameter is valid for Oracle database versions 11.2.0.4 and later and improves the performance of an integrated Replicat.

Each TABLE entry specifies a PDB, schema and table from which to extract data.

Registering Extract – Integrated Extract

In order to enable Integrated Extract mode, the extract must be registered in the database.

To register the Extract to the database for Integrated Extract, issue the following commands on the source Unix system.

```
$(unixserver1) cd /ggs
$(unixserver1) ggsci
GGSCI (unixserver1) > dblogin userid c##ggadmin, password ggadmin
Successfully logged into database CDB$ROOT.

GGSCI (unixserver1) > REGISTER EXTRACT EXTORA DATABASE CONTAINER(PDB1)
Extract EXTORA successfully registered with database at SCN 13317219.
```

The DBLOGIN command establishes a database connection for the specified user. The user is prompted for a password.

The REGISTER command with the **DATABASE** option enables Integrated Capture mode for the Extract group. In this mode, Extract integrates with the database logmining server to receive change data in the form of logical change records (LCR). Extract does not read the redo logs. Extract performs capture processing, filtering, transformation, and other requirements. CONTAINER option allows users to specify a specific PDB or a series of PDBs.

Setting up Extract Checkpoints

Checkpoints enable both Extract and Replicat to process data continuously from one run to another. Checkpoints enable Extract and Replicat to be restarted while ensuring that all records are replicated once and only once.

Extract requires two checkpoints: one into the Oracle redo, which is the source of all database changes, and one into the remote extract trails. Remote extract trails are a series of temporary files created on the target Unix system that contain extracted changes.

To set up these checkpoints on your source system, issue the following commands on Unix.

```
$(unixserver1) cd /ggs
$(unixserver1) ggsci
GGSCI (unixserver1) > ADD EXTRACT EXTORA, INTEGRATED TRANLOG, BEGIN NOW
EXTRACT added.
```

The ADD EXTRACT command establishes an Extract checkpoint group name. The INTEGRATED TRANLOG clause adds this Extract in integrated capture mode. In this mode, Extract integrates with the database logmining server, which passes logical change records (LCR) directly to Extract. Extract does not read the redo log. The BEGIN NOW clause causes Extract to process database operations occurring at or after the time the Extract group was added. Alternatively, you can specify a date and time instead of the keyword NOW.

Defining the Extract Remote Trail Files

To define the extract remote trail on your source system, issue the following command on Unix.

```
GGSCI (unixserver1) > ADD RMTTRAIL ./dirdat/rt, EXTRACT EXTORA, MEGABYTES 10
```

The ADD RMTTRAIL command establishes a checkpoint into a remote extract trail. After each file in this trail reaches approximately 10 megabytes, Extract creates the next file in the sequence. 10 megabytes is used for training purposes only. It is far too small of a value for a production implementation. Files will be named rt000000, rt000001, rt000002 and so on. These files are the source of input to the Replicat program. Note that instead of 'rt' you could specify any two-character prefix.

Choose group names, destination files and sizes appropriate for your environment.

Notes on Remote Extract Trails

Remote extract trail files are only temporary. Manager can be configured (using PURGEOLDEXTRACTS) to delete the trail file when Extract and Replicat are both finished processing it. Therefore, intermediate storage requirements are actually quite small.

Rollover from one file to the next can also be controlled by time of day rather than file size, via the EXTRACT ROLLOVER parameter.

Running Real-time Extract

The Extract program for capturing database changes is initiated from GGSCI on the source Unix system.

```
$(unixserver1) /ggs/ggsci
GGSCI (unixserver1) > START EXTRACT EXTORA
```

Extract continues to run until explicitly stopped via GGSCI.

Creating OGG Replicat User

Integrated replicat cannot connect to the root container. The replicat user must connect directly to a specific PDB. So a specific user must be created.

```
$(unixserver2)  sqlplus / as sysdba
Connect to the specific PDB :

SQL> alter session set container = pdb1 ;
Session altered.

Create the User:

SQL> create user repuser identified by rep_pass container=current ;
User created.

Grant Permissions:

SQL> grant dba to repuser ;
Grant succeeded.

SQL> exec
dbms_goldengate_auth.grant_admin_privilege('REPUSER',container=>'PDB1');
PL/SQL procedure successfully completed.
```

```
Check Connection with new user:

SQL> disconnect
Disconnected from Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Real Application Clusters, Automatic Storage
Management, OLAP, Advanced Analytics and Real Application Testing options

SQL> connect repuser/rep_pass@pdb1
Connected.
```

Configuring the Replicat Parameter File

As with Extract, you must create a Replicat parameter file. Here you can manually use **vi** or another editor on Unix. In this example, we create the following file (called /ggs/dirprm/repora.prm) for this purpose. Make sure the file is saved in a text format. Note that the ordering of the parameters is important.

Alternatively, you can launch the **vi** program within GGSCI on the target Unix system.

```
$(unixserver2) cd /ggs
$(unixserver2) ggsci
GGSCI (unixserver2) > EDIT PARAMS REPORA
--
-- REPLICAT parameter file to replicate changes
-- for TCUSTORD and TCUSTMER.
--
REPLICAT REPORA

ASSUMETARGETDEFS

DISCARDFILE ./dirrpt/repora.dsc, PURGE, MEGABYTES 100

DDL INCLUDE MAPPED
DDLOPTIONS REPORT

DBOPTIONS INTEGRATEDPARAMS(parallelism 6)

USERID repuser@pdb1, PASSWORD rep_pass

MAP pdb1.test.*, TARGET pdb1.test.*;
```

Parameters explained:

REPLICAT REPORA associates this parameter file with the REPORA checkpoint established via GGSCI. This also implicitly establishes the extract trail /ggs/dirdat/rt as the source of data to replicate.

ASSUMETARGETDEFS allows Replicat to assume that the source Oracle tables are structured like the target tables. This eliminates the need to retrieve table definitions from the source system. Source definitions must be generated using DEFGEN if a source table is structured differently than the target.

DISCARDFILE determines where records from operations that fail during replication are output. The discard file is useful for debugging problems during the replication process. This file will contain any rejected rows and the associated causes. Any existing contents are purged at startup when PURGE is specified (APPEND could be specified instead). MEGABYTES specifies the maximum file size.

DDL INCLUDE MAPPED enables apply to the target database of DDL changes for mapped objects.

DDLOPTIONS REPORT enables DDL processing information to be written to the report file.

DBOPTIONS INTEGRATEDPARAMS(parallelism 6) Passes settings for parameters that control the database inbound server within the target Oracle database. Parallelism sets the number of processes.

USERID and PASSWORD must match an existing account in the Oracle database. The active PDB is indicated by @pdb1. In this example the password is in clear text. The best practice is to encrypt the password.

Each MAP entry establishes a relationship between a source Oracle table and a target Oracle table.

Note that no END parameter is specified. The omission of END means that Replicat will continue to run until explicitly stopped by the user (or a fatal error occurs). END is specified when Replicat is run in batch mode versus online mode.

In addition, any replication error will cause Replicat to abort (for example, a duplicate record condition). See the documentation on the following Replicat parameters to customize error response: HANDLECOLLISIONS, OVERRIDEDUPS, INSERTMISSINGUPDATES and REPERROR. Note that restart issues are discussed later in this tutorial.

Setting up a Replicat Checkpoint

The Replicat checkpoint establishes an initial position into the extract trail created by Extract. By default, this will always be the first record extracted. The checkpoint is updated after each transaction, ensuring that all data is processed from run to run.

To set up the Replicat checkpoint, issue the following commands on the target Unix system.

```
GGSCI (unixserver2)> ADD REPLICAT REPORA, INTEGRATED, EXTTRAIL ./dirdat/rt
```

The ADD REPLICAT command establishes the extract trail (EXTTRAIL) created by Extract as the source of information to replicate. REPORA is the name given to this checkpoint group. . The INTEGRATED T clause adds this Replicat in integrated apply mode.

Initial Loading of Oracle Tables utilizing OGG Initial Load Facilities

This section demonstrates the basic features of GoldenGate’s Oracle to Oracle initial load facilities.

Initial Data Extract, Conversion and Load

At this point, data can be extracted directly from the Oracle tables, converted, and then loaded into Oracle tables. Initial load from Oracle to Oracle involves the following tasks.

- Extract is configured as a batch task to retrieve data directly from the source tables
- and send the data directly to a Replicat batch task.
- Replicat is configured as a batch task to populate target tables.

Find Current SCN

The current SCN is collect to determine at what point the real-time replicat should be started and also to have a consistent load state of the target tables. This output will be used in two places: 1) the initial load extract and 2) an option for starting the realtime replicat. This avoids having to use HANDLECOLLISIONS which can be problematic.

```
$(unixserver1)  sqlplus / as sysdba
Connect to the specific PDB :

SQL> alter session set container = pdb1 ;
Session altered.

Determine Current SCN (Save this information):

SQL> SELECT current_scn from v$database ;
CURRENT_SCN
-----------
   75578010
```

Specifying Extract Parameters

Extract parameters are entered into an Extract parameter file you create via any text editor on the source system. The following file (named /ggs/dirprm/initext.prm) is an example of a file created for this purpose. Note that the ordering of the parameters is critical.   Please substitute the highlighted SCN with the SCN that was captured on your system when executing the command above.

```
GGSCI (unixserver1) > EDIT PARAMS INITEXT
--
-- Extract parameter file to capture TCUSTORD
-- and TCUSTMER initial data for Replicat
--
EXTRACT INITEXT

USERID c##ggadmin, PASSWORD ggadmin

RMTHOST unixserver2, MGRPORT 7809

RMTTASK REPLICAT, GROUP INITREP
```

```
TABLE pdb1.test.*, SQLPREDICATE ‘AS OF SCN 75578010’ ;
```

Parameters explained:

Two dashes (--) at the beginning of a line indicates a comment.

EXTRACT is the group name used for this batch task, which retrieves data directly from the tables rather than the redo log and sends the data to the RMTTASK Replicat group.

USERID and PASSWORD must match an existing account in the Oracle database. The active Oracle database is indicated by the user’s ORACLE_SID environment variable. In this example the password is in clear text. The best practice is to encrypt the password.

The RMTHOST parameter specifies the TCP/IP address of the target Unix system to which the data is moved. MGRPORT specifies the well known port number on which Manager has been configured to listen for requests for Extract Server Collectors. RMTHOST can also be specified as a standard TCP/IP host name.

The RMTTASK entry determines where the extracted data is output on the target, in this case, a Replicat group named INITREP.

Each TABLE entry specifies a table from which to extract data, and a TARGET structure for the data. If the source and target column names are the same, no other parameters are required. If your columns have different names, refer to the COLMAP statement on how to explicitly map each column. Make sure you change the schema to match the owner of the table. SQLPREDICATE insures initial load data is consistent to a specified SCN.

Note also that a semi-colon (;) is required at the end of the TABLE entry.

Extracting a Subset of Columns and Rows from a Table

Frequently, a subset of the source table’s columns is required on the target platform, rather than each and every column.

For example, the target application may use only 10 out of the 70 columns in the source table. In this case, the COLMAP clause can limit the mapping to those 10 columns.

As another alternative, the set of rows to transfer can be limited by specifying WHERE clauses. For example, the parameter entry TABLE TCUSTORD, WHERE (CUST_CODE = “ABC” OR CUST_CODE = “XYZ”) would exclude any rows not meeting the criteria.

Configuring the Replicat Parameter File

As with Extract, you must create a Replicat parameter file. Launch GGSCI on the target system and issue the following GGSCI command. (Note that the ordering of the parameters is important) :

```
GGSCI (unixserver2) > EDIT PARAMS INITREP
--
-- REPLICAT parameter file to replicate initial changes
-- for TCUSTMER.
--
REPLICAT INITREP

ASSUMETARGETDEFS
DISCARDFILE ./dirrpt/tcustmer.dsc, APPEND

USERID repuser@pdb1, PASSWORD rep_pass
```

```
        MAP pdb1.test.*, TARGET pdb1.test.*;
```

Parameters explained:

REPLICAT is the batch task group named and must match the name specified in the source Extract parameter file.

ASSUMETARGETDEFS tells Replicat to assume that the source Oracle tables are structured like the target tables. This eliminates the need to retrieve table definitions from the source system. Source definitions must be generated using DEFGEN if a source table is structured differently than the target.

DISCARDFILE determines where operations that fail during replication are output. The discard file is useful for debugging problems during the replication process. It will contain any rejected rows and the associated causes. When PURGE is specified existing contents are purged at startup, or APPEND could be specified instead.

USERID and PASSWORD must match an existing account in the Oracle database.   The active Oracle database is indicated by the user's ORACLE_SID environment variable. In this example the password is in clear text. The best practice is to encrypt the password.

Each MAP entry establishes a relationship between a source Oracle table and a target Oracle table. In this example, we used a wildcard map.

In addition, any replication error will cause Replicat to abort (for example, a duplicate record condition). See the documentation on the following Replicat parameters to customize error response: HANDLECOLLISIONS, OVERRIDEDUPS, INSERTMISSINGUPDATES and REPERROR. Note that restart issues are discussed later in this tutorial.

Adding Extract Batch Task Group

Extract must be defined as a batch task within GGSCI before it may be started. Use GGSCI to configure the Extract Task with the following command on the source system.

```
        GGSCI (unixserver1) > ADD EXTRACT INITEXT, SOURCEISTABLE
```

Adding Replicat Batch Task Group

Replicat must be defined as a batch task within GGSCI before it may be started. Use GGSCI to configure the Replicat Task with the following command on the target system.

```
        GGSCI (unixserver2) > ADD REPLICAT INITREP, SPECIALRUN
```

Running Initial Load

The Extract program is initiated either in online mode with GGSCI or in batch mode from the Unix shell with the following commands on the source system. When Extract is run for the initial load, run it in batch mode as shown below.

```
        GGSCI (unixserver1) > START EXTRACT INITEXT
```

The process may be monitored with the INFO or STATUS commands.

```
GGSCI (unixserver1) > INFO EXTRACT INITEXT
```

Note that the output will be sent to the Extract's and Replicat's report file.

To view the report file, use the following GGSCI command.

```
GGSCI (unixserver1) > VIEW REPORT INITEXT

GGSCI (unixserver2) > VIEW REPORT INITREP
```

When Extract completes, you will see statistics indicating how many records were output into the target tables.

Apply Change Data that was captured during Initial Loading of Data

At this point, you will need to apply the changes that were occurring on the source system while the initial load was being executed.

Initializing the Target While the Source Database Remains On-line

This tutorial utilized the OGG initial load facilities. When the source database is active and you are using OGG tools for doing the initial load, the SCN captured during the initial load step is used to make sure Replicat only processes operations after the as of the load SCN. The AFTERCSN option will avoid duplicate and/or missing record conditions while the replicat is becoming current with the source due to only processing data after the initial load SCN. For the documented best practice of instantiating from an Oracle source, please refer to **Oracle GoldenGate Best Practices: Instantiation from an Oracle Source Database (Doc ID 1276058.1)**

Running Replicat

Replicat is initiated from the target Unix command prompt as follows. Make sure to use the SCN captured in the initial load step. In our example, `75578010` was the SCN captured during the initial load step. Only data captured 'After' SCN `75578010` will be applied.

```
$(unixserver2) cd /ggs

$(unixserver2) ggsci

$GGSCI (unixserver2) > START REPLICAT REPORA, AFTERCSN 75578010
```

Adding Demo DML Transactions

At this point, you can add some demo transactions to the source Oracle database with the following Unix commands on the source system from the OGG Installation directory.

```
$(unixserver1) sqlplus test/test_pass@pdb1

SQL> @demo_ora_misc
```

Operations on the TCUSTMER and TCUSTORD tables will be extracted and replicated.

Adding Demo DDL Operations

At this point, you can add some demo DDL operations to the source Oracle database with the following Unix commands on the source system from the OGG Installation directory.

```
$(unixserver1) sqlplus test/test_pass@pdb1
```

```
SQL> @demo_more_ora_create.sql
```

Creation of the tables MORE_RECS_TBL, EMPOFFICEDTL, EMPPERSONALDTL, EMPSKILLSDTL, EMPPROJECTDTL will be extracted and replicated.

Assessing Replication Status, Diagnosing Problems

Using the REPORT and DISCARD Files

Both Extract and Replicat echo parameters, as well as diagnostic messages, to their respective report files.

In order to view the report file while Extract is running, issue the following command from GGSCI on Unix (EXTORA is the name of the EXTRACT group).

```
GGSCI (unixserver1) > VIEW REPORT EXTORA
```

The Replicat report can be viewed on the target system in a similar manner.

```
GGSCI (unixserver2) > VIEW REPORT REPORA
```

Any errors in the parameter file are output to the report and out files.   Once the message “Run-Time Warnings” appears in the report, all parameters have been validated and data processing has begun.

Discard files also serve as a source for debugging replication problems. Frequently, looking in the discard file at specific record values and error numbers is the fastest path to problem resolution.

Obtaining Extract and Replicat Process Status through GGSCI

To obtain the status and history of an Extract process, issue the following commands.

From GGSCI on Unix

```
GGSCI (unixserver1) > dblogin userid c##ggadmin, password ggadmin
Successfully logged into database CDB$ROOT.


GGSCI (unixserver1) > INFO EXTRACT EXTORA, DETAIL
```

The output of the INFO command will look something like this:

```
EXTRACT    EXTORA    Last Started 2014-12-18 14:56   Status RUNNING
Checkpoint Lag       00:00:04 (updated 00:00:02 ago)
Process ID           30861
Log Read Checkpoint  Oracle Integrated Redo Logs
                     2014-12-18 14:59:54
                     SCN 0.75146836 (75146836)

  Target Extract Trails:

  Trail Name                                       Seqno        RBA     Max
MB Trail Type

  ./dirdat/rt                                          1       1449
10 RMTTRAIL
```

```
Integrated Extract outbound server first scn: 0.75087872 (75087872)

  Extract Source                          Begin             End

  Not Available                           2014-12-18 14:02  2014-12-18 14:59
  Not Available                           2014-12-18 14:02  2014-12-18 14:32
  Not Available                           * Initialized *   2014-12-18 14:02


Current directory    /ggs

Report file          /ggs/dirrpt/EXTORA.rpt
Parameter file       /ggs/dirprm/extora.prm
Checkpoint file      /ggs/dirchk/EXTORA.cpe
Process file         /ggs/dirpcs/EXTORA.pce
Error log            /ggs/ggserr.log
```

The statistics show the history of Extract runs for the EXTORA extract checkpoint group and the current file in the remote extract trail. Also indicated is the current status of the process, in this case RUNNING.

Similarly, to obtain the status and history of a Replicat process, issue the following commands.

From GGSCI on Unix

```
GGSCI (unixserver2) 39> dblogin userid repuser@pdb1 password rep_pass
Successfully logged into database PDB1.

GGSCI (unixserver2) 40> info rep repora, detail

REPLICAT   REPORA    Last Started 2014-12-18 16:49   Status RUNNING
INTEGRATED
Checkpoint Lag       00:00:00 (updated 00:00:03 ago)
Process ID           2085
Log Read Checkpoint  File ./dirdat/rt000002
                     2014-12-18 16:56:21.048066  RBA 3328

INTEGRATED Replicat
DBLOGIN Provided, inbound server name is OGG$REPORA in ATTACHED state


Current Log BSN value: <NULL>


  Extract Source                          Begin             End

  ./dirdat/rt000002                       * Initialized *   2014-12-18 16:56
  ./dirdat/rt000002                       * Initialized *   First Record
  ./dirdat/rt000001                       * Initialized *   First Record
  ./dirdat/rt000000                       * Initialized *   First Record


Current directory    /ggs

Report file          /ggs/dirrpt/REPORA.rpt
Parameter file       /ggs/dirprm/repora.prm
Checkpoint file      /ggs/dirchk/REPORA.cpr
Process file         /ggs/dirpcs/REPORA.pcr
Error log            /ggs/ggserr.log
```

Stopping and Restarting Extract and Replicat

The GGSCI STOP Command

Both Extract and Replicat can be stopped via the GGSCI utility. To stop Extract, run GGSCI from the source system on which Extract is running and issue the following command.

```
GGSCI (unixserver1) > STOP EXTRACT EXTORA
```

This command may take a few seconds to execute.

On the target system, Replicat is stopped with the following command.

```
GGSCI (unixserver2) > STOP REPLICAT REPORA
```

Viewing Replication Statistics

Extract outputs how many inserts, updates and deletes it captured after normal stoppage into the REPORT file. View the REPORT file with the VIEW REPORT command described earlier.

Replicat outputs how many inserts, updates and deletes were applied to the target database in the standard out file and the report file after the process is stopped normally.

Generating interim reports is also possible without stopping Extract and Replicat. See the Extract and Replicat REPORT parameter for more details, and the GGSCI SEND…REPORT commands.

Restarting Extract and Replicat

Both Extract and Replicat are restarted in the same manner as they were originally started, whether or not the previous run ended gracefully or abnormally.

The extraction and replication processes automatically start where they left off, while ensuring no transactions are missed or duplicated. This integrity is guaranteed by the checkpoints maintained by each process.

Where to Go for More Information

Hopefully, this tutorial has provided a quick overview of what you need to do in order to replicate data from Oracle to Oracle. Undoubtedly, you will eventually fine-tune this process in your own environment.

Reference the Oracle Database 12.1 Documentation for additional information on Multitenant.

Reference the Oracle GoldenGate 12c Reference Guide and the Oracle GoldenGate 12c Administration Guide for additional information on:

- Extract Parameters for Windows and Unix
- Replicat Parameters for Windows and Unix

- Extract Management Considerations
- Replicat Management Considerations

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US

blogs.oracle.com/oracle

facebook.com/oracle

twitter.com/oracle

oracle.com

A-Team Chronicles ateam-oracle.com

Hardware and Software, **Engineered to Work Together**

OGG 12c Tutorial for Oracle to Oracle (12c Multitenant Container Database)
January 2015
Author: Tracy West
Contributing Authors:

Oracle is committed to developing practices and products that help protect the environment