

An Oracle White Paper  
March 2015

**Oracle GoldenGate Best Practices:  
Active-Active Configuration with DDL and DML CDR**

Document ID 1574184.1

<b>EXECUTIVE OVERVIEW .....</b>	<b>4</b>
<b>INTRODUCTION.....</b>	<b>4</b>
<b>PREREQUISITES:.....</b>	<b>6</b>
<b>STEPS FOR IMPLEMENTING OGG IN AN ACTIVE-ACTIVE CONFIGURATION.....</b>	<b>7</b>
<b>Create an OGG administrative user account and OGG database user account .....</b>	<b>7</b>
<b>Install OGG Software .....</b>	<b>7</b>
<b>Configure the database to enable DDL Replication.....</b>	<b>7</b>
Enable Supplemental Logging.....	8
Determine What DDL Objects and Schemas will be Filtered & Included .....	9
<b>Additional DDL Parameters .....</b>	<b>10</b>
<b>Parameter Required for Active-Active DML .....</b>	<b>11</b>
<b>Parameters for CDR .....</b>	<b>11</b>
<b>Modify Sequences.....</b>	<b>12</b>
<b>Example Configuration .....</b>	<b>12</b>
Tables used for CDR and DDL Replication Example: .....	13
Manager on both Source and Target:.....	13
Extract on Source:.....	13
Pump on Source:.....	14
Replicat on Source:.....	14
Extract on Target: .....	15
Pump on Target: .....	15
Replicat on Target: .....	16
Create the OGG Processes on Source.....	16
Instantiate the Target Database.....	18
Create the OGG Processes on Target.....	18
<b>Test Scripts for Example Configuration .....</b>	<b>18</b>
Insert Script.....	19
Update Script .....	19
Update Conflict.....	19

Insert Conflict.....	19
Delete Conflict.....	20
<b>CONCLUSION.....</b>	<b>20</b>

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 a highly available 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 examples are provided as is. Oracle consulting service is highly recommended for any customized implementation and a review of the specific production documentation is required

## **Executive Overview**

This document is an introduction to Oracle GoldenGate's best practices and guidelines for configuring Oracle GoldenGate (OGG) in an active-active environment with DDL replication enabled while also using the built in functionality of DML Conflict Detection and Resolution (CDR). This document is intended for Oracle Database Administrators (DBAs), Oracle Developers with some basic knowledge of Oracle GoldenGate software product and Oracle GoldenGate administrators. The document is intended to be an introductory supplement to the existing series of documentation available from Oracle.

The following assumptions have been made during the writing of this document:

- The reader has basic knowledge of Oracle GoldenGate products and concepts
- Referencing Oracle GoldenGate Version 11.2.1+
- Referencing Oracle Version 10.2.0.5 and above
- Referencing OS: All Oracle GoldenGate supported platforms for Oracle

## **Introduction**

Oracle GoldenGate provides data capture from transaction logs and delivery for homogenous and heterogeneous databases and messaging systems. Oracle GoldenGate (OGG) provides a flexible, decoupled architecture that can be used to implement virtually all database replication scenarios.

In an active-active replication environments where the occurrence of conflicts is possible during replication, it is highly desirable to have a conflict management scheme which is enforced as an exception rather than as a norm. Prior to OGG Version 11.2, the resolution if not the detection had to be handled with custom SQL code. Typically, the chances of conflicts occurring during replication are low and hence, checking for conflicts every time before a record is applied on the target is not efficient. However, at the same time it is highly important to ensure that when a conflicting scenario is encountered it is handled in the right manner.

DDL replication is generally not recommended to be setup where DDL is occurring actively on both the source and target database. There currently is no CDR functionality available for DDL. Therefore, DDL should either originate from only one of the systems at a time or DDL operations should not occur on the same objects when issued from the different systems. Otherwise, DDL conflicts could occur and will cause replication to fail due to DDL issues as well as potential DML replication issues.

The goal of this document is to illustrate a generic approach to configuring DDL in an active-active environment as well as enabling the built-in functionality of CDR for conflict detection and resolution of DML operations. By utilizing the built in functionality of CDR, the need for custom programming is reduced if not completely eliminated.

## **Oracle GoldenGate Best Practices: Active-Active Configuration with DDL and DML CDR**

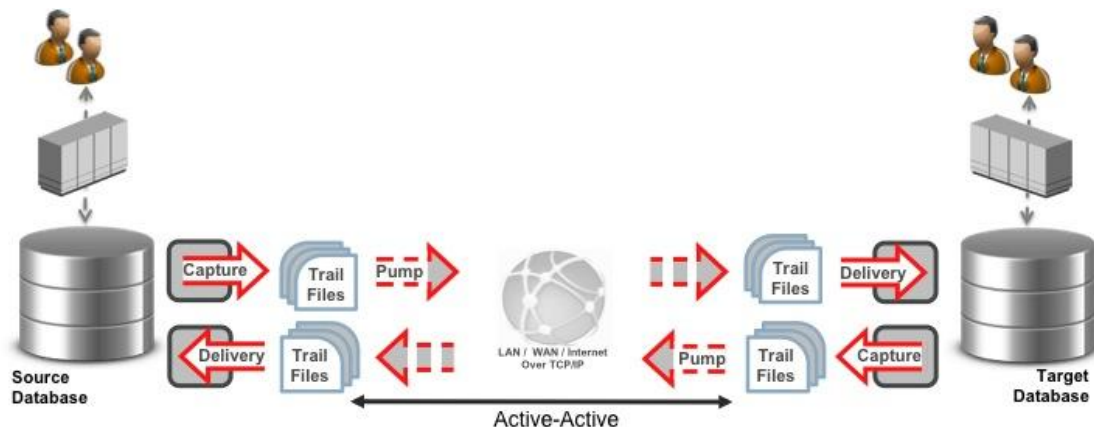
The approach described is the key focus of this document. It illustrates the configuration and the procedure for handling conflicts as an exception using OGG. The sample implementation is provided to illustrate the approach and is specific to the requirements listed in this document and Oracle database. However, depending on the actual requirements and use cases different implementation should most likely be pursued. It is therefore, highly recommended to evaluate the implementation provided in this document from a functional, performance, and maintenance perspectives before adopting it in an actual use case.

## **Prerequisites:**

- OGG Version 11.2+ must be installed on both nodes
- DDL replication must be enabled on both the systems. This step is covered in the installation guide.
- An OGG administrator account created as described in the installation guide.

## Steps for Implementing OGG in an active-active configuration

These steps are required for an active-active environment that will be implementing OGG with DDL and CDR enabled. The various steps will have specific commands as examples. The examples will reference the resources and processes as outlined in the diagram below. This diagram depicts a typical active-active configuration of OGG.



### **Create an OGG administrative user account and OGG database user account**

The OGG Instance will require an OS user account and a database user account. Please follow the instructions in the Oracle GoldenGate Installation Guide for complete details. This must be done on both systems.

### **Install OGG Software**

Please follow the steps as described in the Oracle GoldenGate Installation Guide for the specific software installation steps. This must be done on both systems.

### **Configure the database to enable DDL Replication**

Please follow the steps as described in the Oracle GoldenGate Installation Guide for the specific steps for configuring the database for DDL replication. In OGG versions 11.2 and prior, this requires an outage to install the required objects to support DDL replication. This must be done on both systems. However, if the target system is instantiated with RMAN, this will be configured when the target

database is created from the backup of the source if the source system has been configured for DDL support.

### Enable Supplemental Logging

Minimal supplemental logging needs to be enabled at the database level. It also needs to be enabled at the table level.

To enable minimal supplemental logging at the database level:

```
SQL> alter database add supplemental log data ;
```

Supplemental logging at the table level is critical for replication. But with Conflict, Detection and Resolution (CDR) enabled other fields besides the key column may need to be supplementally logged. This is problematic when active-active DDL is enabled because the table structure and indexes could change. Normally in a DDL replication environment “ADD SCHEMATRANDATA <schema>” is utilized. This ensures that if a table structure or index is modified that the underlying supplemental logging is also automatically updated. If the columns used for CDR will always be logged, ie they are always touched for every type of DML, then supplemental logging should be enabled with “ADD SCHEMATRANDATA”. A good example of this scenario would be a using a column like LAST\_UPDATE\_DATE for CDR that is populated or modified with every Insert/Update.

To enable schema level supplemental logging:

```
GGSCI> ADD SCHEMATRANDATA <schema>
```

If additional columns besides key columns are needed for CDR, and those columns are not always updated as part of the normal DML, then use “ADD TRANDATA <schema>.<table>, COLS ( <CDR col1>, <CDR col2>... )”. The issue with enabling supplemental logging with “ADD TRANDATA” is that if a table structure or PK index changes, the supplemental logging will be modified by extract. Extract will drop the added CDR columns from supplemental logging. So DDL will need to occur during an outage or when DML activity is halted on the database. Also the supplemental logging will also need to be re-added with the CDR columns once the DDL activity is completed to guarantee the CDR columns are also supplementally logged. Otherwise, DDL modifying table structures and indexes will cause replication errors to occur due to supplemental logging losing the required columns for CDR.

To enable table level supplemental logging:

```
GGSCI> ADD TRANDATA <schema>.<table>, COLS ( CDR_COL1, CDR_COL2, ... )
```

If CDR columns are not part of the primary key, and DML needs to occur during DDL changes, then the application must modify supplemental logging as part of the DDL change. If this approach is utilized, do not use the extract parameter “DDLOPTION ADDTRANDATA”. This parameter will



automatically modify the supplemental logging to contain only key columns when table or index structures change.

### Determine What DDL Objects and Schemas will be Filtered & Included

In some environments, not all objects will be replicated. Schemas or specific object types can be excluded from being captured by the DDL trigger itself to reduce the overhead of DDL replication. By default, any DDL occurring on the database will be captured by the DDL trigger. For example, Global Temporary Tables are not supported by OGG. So it is best to exclude these types of objects from the DDL trigger. Also, many applications have multiple schemas. If you are only replicating one of those schemas, it is best to exclude those other schemas from the DDL trigger. To filter DDL at the trigger level, you must use the `ddlaux_addrule()` function. See the OGG installation guide for more detailed information on this function. Using this function does not require an outage.

To exclude the specific schema “BEE\_MDS” and Global Temporary Tables from being captured execute:

```
SQL> @ddlaux_addrule.sql

Contents of ddlaux_addrule.sql:
  BEGIN
  DECLARE
    sno NUMBER;
  BEGIN
    /* Exclude Schema BEE MDS */
    sno := oragg.ddlaux_addrule(owner name => 'BEE MDS');
    /* Exclude Temporary Tables */
    sno := oragg.ddlaux_addrule(base_obj_property => ddlaux.TB_TEMP, obj_type =>
ddlaux.TYPE_TABLE);
  END;
  END;
  /
```

More granular filtering maybe required at the parameter level. Some objects require `sysdba` permissions to be executed by the replicat. “ALTER SYNONYM”, “GRANT ON DIRECTORY” are common commands that will fail on the replicat because the OGG user does not have `sysdba` permission. DDL on Public Synonyms also requires `sysdba`. These objects can be excluded in the extract using the `DDL EXCLUDE` parameter.

To exclude at the extract level add to parameter file:

```
DDL
EXCLUDE MAPPED OBJTYPE 'SYNONYM' OPTYPE ALTER &
EXCLUDE OPTYPE GRANT INSTR 'on directory' &
EXCLUDE OPTYPE GRANT INSTR 'ON DIRECTORY'
```

In most cases, you will want to extract both `MAPPED` and `UNMAPPED` DDL on the schema that you are replicating. For a deeper explanation of `MAPPED` and `UNMAPPED` parameters, please refer to the OGG Installation Guide and OGG Reference Guide.

To include at the extract level mapped and unmapped DDL for the DEMO schema:

```
DDL
INCLUDE MAPPED &
INCLUDE UNMAPPED OBJNAME DEMO.* &
```

### **Additional DDL Parameters**

Some additional parameters are required for easier troubleshooting of DDL issues, managing DDL data, updating metadata and updating of supplemental logging.

For easier troubleshooting in both extract and replicat:

```
--- This parameter reports DDL being replicated.
DDLOPTIONS REPORT
```

Once a table or key index is altered, it is usually required to also update the supplemental logging of a given table. This is done without locking and automatically if ADD SCHEMATRANDATA was used to enable supplemental logging. If ADD TRANDATA was used to enable supplemental logging, the extract must update the supplemental logging. This is done with the DDLOPTIONS ADDTRANDATA parameter. Due to timing and locking issues, it is recommended to not have DML occurring at the same time as DDL modifications when using ADD TRANDATA. If DDL and DML must occur concurrently, then ADD SCHEMATRANDATA should be used to enable supplemental logging to avoid any replication issues. Also any columns used by CDR will either need to be part of the key column or a column always modified by the application.

For updating supplemental logging when ADD TRANDATA is used in the extract:

```
--- Not Required if Using ADDSCHEMATRANDATA SCOTT
--- DDLOPTIONS ADDTRANDATA
```

In an active-active configuration, each Replicat must be configured to update its object metadata cache whenever the remote Extract sends over a captured Replicat DDL statement. To satisfy this requirement, use the UPDATEMETADATA option in the DDLOPTIONS statement in the Replicat parameter files on both systems. The extract parameter file on both systems should also include the GETREPLICATES option in the DDLOPTIONS statement.

The resultant DDLOPTIONS statements should look as follows:

```
Extract (primary and secondary)
DDLOPTIONS GETREPLICATES
Replicat (primary and secondary)
DDLOPTIONS UPDATEMETADATA
```

While DDL is being extracted, the DDL objects containing that data will need to be periodically purged of old data to keep the objects storage requirements within reason. Much like the manager is responsible for purging old trail data, the manager must also purge old DDL data.

For purging old DDL data include in the manager parameter file:

```
--- Maintain DDL objects
USERID oragg, PASSWORD <encrypted password>, ENCRYPTKEY <keyname>
PURGEDDLHISTORY MINKEEPDAYS 30 MAXKEEPDAYS 60, FREQUENCYHOURS 24
PURGEMARKERHISTORY MINKEEPDAYS 30 MAXKEEPDAYS 60, FREQUENCYHOURS 24
```

### **Parameter Required for Active-Active DML**

Active-Active DML replication is fairly straight forward with OGG. The extract just needs to be aware of the replicat database user in order exclude that activity.

To exclude replicat activity add this parameter to the extract:

```
TRANLOGOPTIONS EXCLUDEUSER <replicat username>
```

### **Parameters for CDR**

Both Extract and Replicat will require parameters for CDR. For Extract, the before image information being extracted needs to be defined. For Replicat, the conflict rules and resolution rules need to be defined. Please refer to the OGG Administrative Guide for the many options for these parameters.

In Extract, either CDROPTIONS with `_LOGALLSUPPCOL` option or `GETUPDATEBEFORES` is required to collect the before image for Conflict Detection. If using `CDROPTIONS`, use `GETBEFORECOLS` to define what columns are required. Whatever columns are defined in `GETBEFORECOLS`, those columns must be supplementally logged. Using the `ALL` option for `GETBEFORECOLS` will capture only the columns that are supplementally logged. `ALL` does not mean, that the before image of `ALL` columns will be extracted unless all columns are actually supplementally logged. `NOCOMPRESSDELETES` will capture the full delete row. Otherwise only key columns will be captured for deletes.

Extract:

```
CDROPTIONS LOGALLSUPPCOLS
--- Turn on NOCOMPRESSDELETES to capture the full delete row. Otherwise only key
--- columns will be captured for deletes.
NOCOMPRESSDELETES
--- GETBEFORECOLS is required when using CDROPTIONS. In this case,
--- ALL supplementally logged columns are extracted.
TABLE demo.demo users,
GETBEFORECOLS (
ON UPDATE ALL,
ON DELETE ALL);
```

In Replicat, the map statement needs to define 1) the columns used to determine a conflict (`COMPARECOLS`), 2) the type of conflict (`INSERTROWEXISTS`, `UPDATEROWEXISTS`, `UPDATEROWMISSING`, `DELETEROWEXISTS`, `DELETEROWMISSING`) and 3) how to resolve the conflict. The conflict resolution (`RESOLVECONFLICT`) has many options. Please refer the OGG Administrative and Reference Guide for more examples and options.

Replicat:

```
MAP demo.demo_users, TARGET demo.demo_users,
```

```
RESOLVECONFLICT (INSERTROWEXISTS,  
(DEFAULT, USEMAX(CREATION DATE))),  
RESOLVECONFLICT (UPDATEROWEXISTS,  
(DEFAULT, USEMAX(LAST_UPD_DATE))),  
RESOLVECONFLICT (UPDATEROWMISSING,  
(DEFAULT, DISCARD)),  
RESOLVECONFLICT (DELETEROWEXISTS,  
(DEFAULT, DISCARD)),  
RESOLVECONFLICT (DELETEROWMISSING,  
(DEFAULT, DISCARD)),  
COMPARECOLS (  
ON UPDATE, KEYINCLUDING(LAST_UPD_DATE),  
ON DELETE, ALL));  
  
MAP demo.demo_users, target demo.demo_users_exp,  
colmap (USEDEFAULTS,  
op type=@GETENV("GGHEADER", "OPTYPE"),  
op time=@GETENV("GGHEADER", "COMMITTIMESTAMP")),  
EXCEPTIONSONLY,  
INSERTALLRECORDS;
```

### **Modify Sequences**

Sequences should never be replicated in an active-active environment. They are usually used to create a unique identifier for a row. In an active-active environment, one system should set sequences to odd values and the other system should set sequences to even values. This insures that conflicts on the generated values will not occur. Well before OGG processes are created and the instantiation of the target begins, sequences on the source should be modified to either be only even values or odd values.

Once the instantiation of the target is completed, sequences should be modified to be the opposite of the source. For example, if sequences were modified to be even on the source then the target sequences should be modified to be odd.

In a multi-master environment, sequences are incremented by the number of systems that comprise the multi-master. For example if 3 systems make up a given multi-master environment, then each system would increment by 3. So if the current sequence number is 100, then System1 sequence would be altered to start with 101 and increment by 3, System2 sequence would be altered to start with 102 and increment by 3, System3 sequence would be altered to start with 103 and increment by 3.

### **Example Configuration**

This example is an active-active configuration for DDL and DML. The CDR columns being utilized are CREATION\_DATE which is populated on an insert and LAST\_UPD\_DATE which is modified on update operations. CREATION\_DATE will be used determine and resolve an INSERT COLLISION, and LAST\_UPD\_DATE will be used to determine and resolve an UPDATE COLLISION. For DELETE COLLISIONS the full record will be used to determine a collision.

The resolution logic is that if a collision occurs then the record with the most current CREATION\_DATE for inserts and LAST\_UPD\_DATE for updates should be used. For DELETES if the record doesn't match, then the entire record is put in the exception table and the record is not deleted from the database.

## Tables used for CDR and DDL Replication Example:

```
CREATE TABLE "DEMO"."DEMO_USERS"  
  (  
    "ID" NUMBER,  
    "FIRST_NAME" VARCHAR2(30),  
    "LAST_NAME" VARCHAR2(30),  
    "CREATION_DATE" TIMESTAMP (6),  
    "LAST_UPD_DATE" TIMESTAMP (6),  
    CONSTRAINT "PK_DEMO" PRIMARY KEY ("ID") )  
  TABLESPACE "USERS";  
  
CREATE TABLE "DEMO"."DEMO_USERS_EXP"  
  (  
    "ID" NUMBER,  
    "FIRST_NAME" VARCHAR2(30),  
    "LAST_NAME" VARCHAR2(30),  
    "CREATION_DATE" TIMESTAMP (6),  
    "LAST_UPD_DATE" TIMESTAMP (6),  
    "OP_TYPE" VARCHAR2(20),  
    "OP_TIME" TIMESTAMP (6)  
  )  
  TABLESPACE "USERS";
```

## Manager on both Source and Target:

```
PORT 7801  
--- Lockdown ports used by OGG  
DYNAMICPORTLIST 7802-7809  
--- Maintain DDL objects  
USERID gg admin, PASSWORD <encrypted password>, ENCRYPTKEY <keyname>  
PURGEDDLHISTORY MINKEEPDAYS 30 MAXKEEPDAYS 60, FREQUENCYHOURS 24  
PURGEMARKERHISTORY MINKEEPDAYS 30 MAXKEEPDAYS 60, FREQUENCYHOURS 24  
--- Maintain Trail files  
PURGEOLDEXTRACTS ./* , USECHECKPOINTS  
--- Performance and Alert Parameters  
AUTORESTART ER *, RETRIES 3, WAITMINUTES 10, RESETMINUTES 60  
DOWNREPORTMINUTES 15  
DOWNCRITICAL  
LAGCRITICALSECONDS 10  
LAGINFOMINUTES 0  
LAGREPORTMINUTES 15
```

## Extract on Source:

```
EXTRACT e a conf  
USERID gg admin, PASSWORD gg admin  
exttrail ./dirdat/ea  
--- DDL Parameters  
DDL INCLUDE MAPPED &  
  INCLUDE UNMAPPED OBJNAME DEMO.* &  
  EXCLUDE MAPPED OBJTYPE 'SYNONYM' OPTYPE ALTER &  
  EXCLUDE OPTYPE GRANT INSTR 'on directory' &  
  EXCLUDE OPTYPE GRANT INSTR 'ON DIRECTORY'  
--- This parameter reports DDL being replicated.  
--- This is very useful for troubleshooting.  
DDLOPTIONS REPORT  
  
--- Capture DDL changes so that the target replicat can update metadata  
DDLOPTIONS GETREPLICATES
```

## Oracle GoldenGate Best Practices: Active-Active Configuration with DDL and DML CDR

```
--- Not Required if Using ADDSCHEMATRANDATA demo
--- DDLOPTIONS ADDTRANDATA
--- Classic Extract reading ASM log files
TRANLOGOPTIONS DBLOGREADER
--- Exclude Replicat Activity
TRANLOGOPTIONS EXCLUDEUSER gg_admin
--- Grab Before Images to supplementally logged columns
CDROPTIONS LOGALLSUPPCOLS
--- Grab All the Columns for a Delete
NOCOMPRESSDELETES

--- Table we are replicating and define what before images are required.
--- In this case we want the before image of all columns that are logged for
--- UPDATES and DELETES.

TABLE demo.demo users,
GETBEFORECOLS (
ON UPDATE ALL,
ON DELETE ALL );
```

### Pump on Source:

```
EXTRACT p a conf
PASSTHRU
RMTHOST <Target Hostname>, MGRPORT 7801
RMTTRAIL ./dirdat/ra
TABLE demo.*;
```

### Replicat on Source:

```
REPLICAT r a conf
USERID gg_admin, PASSWORD gg_admin

ASSUMETARGETDEFS
DISCARDFILE ./dirrpt/r a conf.dsc APPEND

--- Defer Triggers and Constraints, to avoid issues with
--- delete cascade constraints and data created by triggers
DBOPTIONS DEFERRERCONST
DBOPTIONS SUPPRESSTRIGGERS

--- DDL parameters
DDL INCLUDE ALL
DDLOPTIONS REPORT

--- Update metadata after DDL changes have been applied to source
DDLOPTIONS UPDATEMETADATA

--- Set Conflict Rules and Resolution
--- USEMAX will force the most current row to be used on a conflict
--- for an insert or update.
--- If a row is missing on an update or delete,
--- then the record is written to the exception table.
--- If a row exists on an Delete but it doesn't match all the columns,
--- then the record is written to the exception table.
MAP demo.demo_users, TARGET demo.demo_users,
RESOLVECONFLICT (INSERTROWEXISTS,
(DEFAULT, USEMAX(CREATION DATE))),
RESOLVECONFLICT (UPDATEROWEXISTS,
(DEFAULT, USEMAX(LAST UPD DATE))),
RESOLVECONFLICT (UPDATEROWMISSING,
(DEFAULT, DISCARD)),
```

## Oracle GoldenGate Best Practices: Active-Active Configuration with DDL and DML CDR

```
RESOLVECONFLICT (DELETEROWEXISTS,  
(DEFAULT, DISCARD)),  
RESOLVECONFLICT (DELETEROWMISSING,  
(DEFAULT, DISCARD)),  
COMPARECOLS (  
ON UPDATE, KEYINCLUDING (LAST_UPD_DATE),  
ON DELETE, ALL));  
  
--- Conflicts are written to the exception table with the optype and commit timestamp  
MAP demo.demo users, target demo.demo users exp,  
colmap (USEDEFAULTS,  
op type=@GETENV("GGHEADER", "OPTYPE"),  
op time=@GETENV("GGHEADER", "COMMITTIMESTAMP")),  
EXCEPTIONSONLY,  
INSERTALLRECORDS;
```

### Extract on Target:

```
EXTRACT e_b_conf  
USERID gg admin, PASSWORD gg admin  
exttrail ./dirdat/eb  
--- DDL Parameters  
DDL INCLUDE MAPPED &  
    INCLUDE UNMAPPED OBJNAME DEMO.* &  
    EXCLUDE MAPPED OBJTYPE 'SYNONYM' OPTYPE ALTER &  
    EXCLUDE OPTYPE GRANT INSTR 'on directory' &  
    EXCLUDE OPTYPE GRANT INSTR 'ON DIRECTORY'  
--- This parameter reports DDL being replicated.  
--- This is very useful for troubleshooting.  
DDLOPTIONS REPORT  
  
--- Capture DDL changes so that the target replicat can update metadata  
DDLOPTIONS GETREPLICATES  
  
--- Not Required if Using ADDSCHEMATRANDATA demo  
--- DDLOPTIONS ADDTRANDATA  
--- Classic Extract reading ASM log files  
TRANLOGOPTIONS DBLOGREADER  
--- Exclude Replicat Activity  
TRANLOGOPTIONS EXCLUDEUSER gg admin  
--- Grab Before Images to supplementally logged columns  
CDROPTIONS _LOGALLSUPPCOLS  
--- Grab All the Columns for a Delete  
NOCOMPRESSDELETES  
  
--- Table we are replicating and define what before images are required.  
--- In this case we want the before image of all columns that are logged for  
--- UPDATES and DELETES.  
  
TABLE demo.demo users,  
GETBEFORECOLS (  
ON UPDATE ALL,  
ON DELETE ALL );
```

### Pump on Target:

```
EXTRACT p_b_conf  
PASSTHRU  
RMTHOST <Source Hostname>, MGRPORT 7801  
RMTTRAIL ./dirdat/ra  
TABLE demo.*;
```

## Replicat on Target:

```
REPLICAT r_b_conf
USERID gg admin, PASSWORD gg admin

ASSUMETARGETDEFS
DISCARDFILE ./dirrpt/r b conf.dsc APPEND

--- Defer Triggers and Constraints, to avoid issues with
--- delete cascade constraints and data created by triggers
DBOPTIONS DEFERRERCONST
DBOPTIONS SUPPRESSTRIGGERS

--- DDL parameters
DDL INCLUDE ALL
DDLOPTIONS REPORT

--- Update metadata after DDL changes have been applied to source
DDLOPTIONS UPDATEMETADATA

--- Set Conflict Rules and Resolution
--- USEMAX will force the most current row to be used on a conflict
--- for an insert or update.
--- If a row is missing on an update or delete,
--- then the record is written to the exception table.
--- If a row exists on an Delete but it doesn't match all the columns,
--- then the record is written to the exception table.
MAP demo.demo users, TARGET demo.demo users,
RESOLVECONFLICT (INSERTROWEXISTS,
(DEFAULT, USEMAX(CREATION DATE))),
RESOLVECONFLICT (UPDATEROWEXISTS,
(DEFAULT, USEMAX(LAST_UPD_DATE))),
RESOLVECONFLICT (UPDATEROWMISSING,
(DEFAULT, DISCARD)),
RESOLVECONFLICT (DELETEROWEXISTS,
(DEFAULT, DISCARD)),
RESOLVECONFLICT (DELETEROWMISSING,
(DEFAULT, DISCARD)),
COMPARECOLS (
ON UPDATE, KEYINCLUDING(LAST_UPD_DATE),
ON DELETE, ALL));

--- Conflicts are written to the exception table with the optype and commit timestamp
MAP demo.demo_users, target demo.demo_users_exp,
colmap (USEDEFAULTS,
op type=@GETENV("GGHEADER", "OPTYPE"),
op time=@GETENV("GGHEADER", "COMMITTIMESTAMP")),
EXCEPTIONSONLY,
INSERTALLRECORDS;
```

## Create the OGG Processes on Source



## Oracle GoldenGate Best Practices: Active-Active Configuration with DDL and DML CDR

1. Enable Supplemental logging on source database tables. Make sure the ORACLE\_SID is set in the environment and minimal supplemental logging enabled in the database. ADD SCHEMATRANDATA is the best practice approach when doing DDL replication.

```
GGSCI> DBLOGIN USERID gg_adm PASSWORD gg_adm
GGSCI> ADD SCHEMATRANDATA DEMO
Or
GGSCI> ADD TRANDATA DEMO.DEMO_USER
```

2. Create Manager using Parameter file on Source and Target

3. Create Extract Process on Source

- a. Register Extract with database for Integrated Extract. Make sure the ORACLE\_SID is set in the environment.

```
GGSCI> DBLOGIN USERID gg_adm PASSWORD gg_adm
GGSCI> REGISTER EXTRACT e_a_conf DATABASE
```

- b. Create Extract

```
For Integrated Extract
GGSCI> ADD EXTRACT e_a_conf, INTEGRATED TRANLOG, BEGIN NOW
```

```
For Classic Extract
GGSCI> ADD EXTRACT e_a_conf, TRANLOG, BEGIN NOW
```

```
Create Local Trail File
GGSCI> ADD EXTTRAIL ./dirdat/ea, EXTRACT e_a_conf, MEGABYTES 100
```

4. Create Extract Pump on Source

```
GGSCI> ADD EXTRACT p_a_conf, exttrailsource ./dirdat/ea
```

```
Create Remote Trail File
GGSCI> ADD RMTTRAIL ./dirdat/ra, EXTRACT p_a_conf, MEGABYTES 100
```

5. Create Replicat on Source . The checkpoint table is critical to insure that data is not replayed in the event of a failover.

```
GGSCI> DBLOGIN USERID gg_adm, PASSWORD gg_adm
GGSCI> ADD REPLICAT r_b_conf, exttrail ./dirdat/rb, CHECKPOINTTABLE ggckpt
```

## Instantiate the Target Database

This topic in and of itself is a full white paper. Please refer to document 1276058.1 in metalinks for more information on this step. Since this is an example, the table is empty. So you can just create the table in the target database.

## Create the OGG Processes on Target

### 1. Create Extract Process on Target

- a. Register Extract with database for Integrated Extract. Make sure the ORACLE\_SID is set in the environment.

```
GGSCI> DBLOGIN USERID gg_adm PASSWORD gg_adm
GGSCI> REGISTER EXTRACT e_b_conf DATABASE
```

- b. Create Extract

```
For Integrated Extract
GGSCI> ADD EXTRACT e_b_conf, INTEGRATED TRANLOG, BEGIN NOW
```

```
For Classic Extract
GGSCI> ADD EXTRACT e_b_conf, TRANLOG, BEGIN NOW
```

```
Create Local Trail File
GGSCI> ADD EXTTRAIL ./dirdat/eb, EXTRACT e_b_conf, MEGABYTES 100
```

### 2. Create Extract Pump on Target

```
GGSCI> ADD EXTRACT p_b_conf, exttrailsource ./dirdat/eb
```

```
Create Remote Trail File
GGSCI> ADD RMTTRAIL ./dirdat/rb, EXTRACT p_b_conf, MEGABYTES 100
```

3. Create Replicat on Target. The checkpoint table is critical to insure that data is not replayed in the event of a failover.

```
GGSCI> DBLOGIN USERID gg_adm, PASSWORD gg_admin
GGSCI> ADD REPLICAT r_a_conf, exttrail ./dirdat/ra, CHECKPOINTTABLE ggckpt
```

## Test Scripts for Example Configuration

The following scripts have been included to demonstrate some of the various conflicts. The create conflict script uses the user gg\_admin to generate the conflicts on the source without it being

replicated to the target due to the EXCLUDEUSER parameter. The generate conflict scripts use the user demo which will be replicated across. These are all shell scripts.

### Insert Script

```
#!/bin/sh
sqlplus -s "/ as sysdba"<<EOFF
insert into demo.demo_users values (1, 'FIRST', 'LAST', sysdate, sysdate);
commit;
exit
EOFF
```

### Update Script

```
#!/bin/sh
sqlplus -s "demo/demo"<<EOFF
update demo.demo_users
set first_name='FIRST_NAME_UPD',
last_name='LAST_NAME_UPD',
last_upd_date=sysdate+1 WHERE id=1 ;
commit;
exit
EOFF
```

### Update Conflict

This update conflict will cause a conflict to occur, but the record will be updated on the target because the last\_upd\_date is more current.

Create update conflict

```
#!/bin/sh
sqlplus -s "gg admin/gg admin"<<EOFF
update demo.demo users
set last_upd_date=sysdate - 2 WHERE id=1;
commit;
```

Generate update conflict

```
#!/bin/sh
sqlplus -s "demo/demo"<<EOFF
update demo users
set first_name='FIRST_NAME_UPD CON H',
last_name='LAST_NAME_UPD_CON_H',
last_upd_date=sysdate + 5 WHERE id=1;
commit;
exit
EOFF
```

### Insert Conflict

There are 2 insert conflicts here. You will have to delete the row on the source before you can generate each type of insert conflict. The low insert conflict will go to the exception table because the CREATE\_DATE in the target is more current than the row being applied. The high insert conflict will over write the record on the target because the CREATE\_DATE is more current than the CREATE\_DATE on the target.

Create Insert conflict

```
#!/bin/sh
```

## Oracle GoldenGate Best Practices: Active-Active Configuration with DDL and DML CDR

```
sqlplus -s "gg_admin/gg_admin"<<EOFF
delete from demo.demo users;
commit;
exit
EOFF
```

Generate Low Insert conflict

```
#!/bin/sh
sqlplus "demo/demo"<<EOFF
insert into demo.demo users values (1, 'FIRST CON L','LAST CON L',sysdate-1,sysdate-1);
commit;
exit
EOFF
```

Generate High Insert conflict

```
#!/bin.sh
sqlplus "demo/demo"<<EOFF
insert into demo.demo_users values (1, 'FIRST_CON_H','LAST_CON_H',sysdate+1,sysdate+1);
commit;
exit
EOFF
```

## Delete Conflict

This delete conflict will not delete the record on the target because the delete record will not match the record on the target.

Create delete conflict

```
#!/bin/sh
sqlplus -s "gg admin/gg admin"<<EOFF
update demo.demo users
set last_name='LAST_NAME_DEL_EXISTS' WHERE id=1;
commit;
exit
EOFF
```

Generate delete conflict

```
#!/bin/sh
sqlplus -s "demo/demo"<<EOFF
delete from demo users WHERE id=1;
commit;
exit
EOFF
```

## Conclusion

This paper has presented many different OGG features working together to provide a comprehensive Active-Active solution with DDL enabled. This paper is intended to address a basic active-active implementation of DDL with CDR enabled for DML. Your implementation may require a more complex configuration of CDR rules. Please refer to the OGG Administrative Guide for a more extensive view of CDR.

## Oracle GoldenGate Best Practices: Active-Active Configuration with DDL and DML CDR



Oracle GoldenGate Best Practices: Active-Active Configuration with DDL and DML CDR  
March 2015

Author:s Tracy West, Sourav Bhattacharya  
Contributing Authors:

Oracle Corporation  
World Headquarters  
500 Oracle Parkway  
Redwood Shores, CA 94065  
U.S.A.

Worldwide Inquiries:  
Phone: +1.650.506.7000  
Fax: +1.650.506.7200  
oracle.com



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

Copyright © 2012, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. UNIX is a registered trademark licensed through X/Open Company, Ltd. 0410

**SOFTWARE. HARDWARE. COMPLETE.**