# How To Replicate Data from Oracle to Postgres Using GoldenGate (Doc ID 1544137.1)

**In this Document**



## APPLIES TO:

Oracle GoldenGate - Version 11.2.1.0.0 and later
Information in this document applies to any platform.

## GOAL

Starting with Oracle GoldenGate 11.2.1.0.2 data can now be replicated between Oracle and Postgres. This note contains a basic setup how to replicate data between Oracle and Postgres which can be easily modified to fit your needs.

## SOLUTION

To replicate data between an Oracle and a PostgreSQL database a GoldenGate installation for both databases is required. In this example the host where the Oracle database set up is done is called SOURCE and the machine with the PostgreSQL database is called TARGET. In addition to keep it simple no data pump process is configured. Instead the extract process writes the data to the target machine and the replicat process reads the extract file directly.

### 0. Installing and setting up Oracle GoldenGate connecting to an Oracle database

Before you install the GoldenGate software please make sure the following environment variables are set and point to your Oracle database installation:

```
ORACLE_HOME
ORACLE_SID
LD_LIBRARY_PATH
```

Also please make sure the Oracle database is in archive log mode. If the source database is not in archivelog mode, enable it using the steps below:

```
alter system set log_archive_dest='LOCATION=USE_DB_RECOVERY_FILE_DEST' scope=both sid='*';
shutdown immediate
startup mount
alter database archivelog;
alter database open;
```

In addition you should enable minimum supplemental logging:

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

Once the Oracle database is prepared you can start installing GoldenGate. When the software was downloaded from Oracle's software delivery cloud (https://edelivery.oracle.com) or from "My Oracle Support portal" please place the downloaded file into the directory you want to designate as GoldenGate home, unzip the file and inflate it using the tar command.

Now add the GoldenGate directory to your LD_LIBRARY_PATH and PATH:

```
export PATH=$PATH:/home/oracle/ggs
export LD_LIBRARY_PATH=$ORACLE_HOME/lib:/home/oracle/ggs/lib
```

The first step is to open the GoldenGate command line interface and to create the necessary subdirectories:

```
GGSCI (SOURCE.ORACLE.COM) 1> create subdirs

Creating subdirectories under current directory /home/oracle/ggs

Parameter files                /home/oracle/ggs/dirprm: already exists
Report files                   /home/oracle/ggs/dirrpt: created
Checkpoint files               /home/oracle/ggs/dirchk: created
Process status files           /home/oracle/ggs/dirpcs: created
SQL script files               /home/oracle/ggs/dirsql: created
Database definitions files     /home/oracle/ggs/dirdef: created
Extract data files             /home/oracle/ggs/dirdat: created
Temporary files                /home/oracle/ggs/dirtmp: created
Stdout files                   /home/oracle/ggs/dirout: created
```

The second step is to create a parameter file for the manager which at least contains a PORT number for the manager:

```
GGSCI (SOURCE.ORACLE.COM) 2> edit param mgr
```

add the following content to the parameter file:

```
PORT 7809
```

Save the parameter file, exit from the editor, start the manager and verify if it is running:

```
GGSCI (SOURCE.ORACLE.COM) 3> start mgr

GGSCI (SOURCE.ORACLE.COM) 4> info all

Program     Status      Group       Lag at Chkpt  Time Since Chkpt
```

```
MANAGER     RUNNING
```

### 1. Installing and setting up Oracle GoldenGate on the Postgres machine

The installation of GoldenGate on the Postgres machine is similar to the installation on the Oracle box. Get the download file and then unzip and untar it in a directory you want to use as GoldenGate Home directory. Then add the lib directory to the LD_LIBRARY_PATH.

```
mkdir ggs
cd ggs
unzip V34006-01.zip
tar xvf *.tar
[oracle@TARGET ggs]$ export LD_LIBRARY_PATH=/home/oracle/ggs/lib
```

GoldenGate uses an ODBC connection to connect to the Postgres database. The ODBC driver is shipped with the installation and on Unix you have to create the ODBC configuration file which is commonly called odbc.ini on your own.

The odbc.ini file is similar to an address book for the odbc driver. It is located by default in the ODBC_HOME directory, but can be placed anywhere you like.

A side note how odbc works: The odbc driver (nothing else than a library) gets a request to connect to a server described in the odbc.ini file. The alias for the description of the server is called:
Data Source Name (=DSN).
Then the driver reads the information from the odbc.ini file according to the specified DSN and connects to the server.

The prerequisite for the odbc driver to connect to the server is the configured odbc.ini.
It is divided into 3 different sections:

```
[ODBC Data Sources]
[<DSN>]
[ODBC]
```

[ODBC Data Source] is the section that contains all the available DSNs.
[<DSN>] contains the different names of the DSNs and specifies the connect details.
[ODBC] is the general section for the odbc driver

There's an example in the GoldenGate Postgres installation guide:

```
[ODBC Data Sources]
postgre=DataDirect 6.1 PostgreSQL Wire Protocol
[ODBC]
IANAAppCodePage=4
InstallDir=/home/fin/fin13004/postgres/v11201_120402
[postgre]
Driver=/home/fin/fin13004/postgres/v11201_120402/lib/GGpsql25.so
Description=DataDirect 6.1 PostgreSQL Wire Protocol
Database=fin
HostName=12.345.6.789
PortNumber=5432
LogonID=postgres
Password=postgre
```

Although most of the parameters are self explaining a few words to the odbc.ini file.

The section [ODBC Data Sources] contains in general a list of available data sources (which you can name as you want) and the sample from the manual has one data source called postgre. The configuration behind the data source postgre is found in the section [postgre].
It contains the hostname and port of the Postgres server, the Postgres database itself, the driver library being used and the user id and password of the remote database server (LogonID and password are not required, they can be specified also in the replicat parameter file).

The [ODBC] section contains general parameters like a code page specification.

As not everybody might be experienced configuring ODBC DSNs let's create an odbc.ini step by step. As we know we first define the [ODBC Data Sources] section with a DSn name of our choice, then create the [ODBC] section

```
[ODBC Data Sources]
<a name of your choice used as alias for the ODBC connection>=DataDirect 6.1 PostgreSQL Wire Protocol
[ODBC]
```

These are generic settings which you can simply copy/paste and where you have to replace <a name of your choice used as alias for the ODBC connection> with any name of your choice, for example GG_Postgres.

```
IANAAppCodePage=4
```

is being used for national language support. The value 4 represents the ISO-8859-1 character set, 106 a Unicode UTF8 character set. The setting should always reflect the character set of the Postgres database. More details about the setting are covered in another note (1543702.1)

```
 InstallDir=/home/oracle/ggs
```

Make sure to change the path to the location of your GoldenGate installation directory.

```
[<a name of your choice used as alias for the ODBC connection matching the value in the ODBC Data
Sources section>]
```

This will be the ODBC DSN (Data Source Name) similar to the Oracle tns name containing all necessary details to connect to the Postgres database. It's name should match the value in the [ODBC Data Sources] section. Next configuration parameter is the ODBC driver library. Just make sure to replace <your goldengate home directory> with your GoldenGate home directory:

```
Driver=<your goldengate home directory>/lib/GGpsql25.so
```

The parameter

```
Description=DataDirect 6.1 PostgreSQL Wire Protocol
```

is not really needed, so just copy/paste it.

At the end we have to specify the details for your Postgres database:

```
Database=<your Postgres Database>
HostName=<the hostname of the Postgres database>
PortNumber=<the port number of the Postgres database>
LogonID=<a username of the Postgres database>
Password=<a password of the Postgres database>
```

Here the file I used:

```
[oracle@TARGET ggs]$ vi odbc.ini
[ODBC Data Sources]
GG_Postgres=DataDirect 6.1 PostgreSQL Wire Protocol
```

```
[ODBC]
IANAAppCodePage=106
InstallDir=/home/oracle/ggs
[GG_Postgres]
Driver=/home/oracle/ggs/lib/GGpsql25.so
Description=DataDirect 6.1 PostgreSQL Wire Protocol
Database=GGTest
HostName=TARGET.ORACLE.COM
PortNumber=5432
LogonID=postgres
Password=postgres
```

Finally we need to export an ODBC environment variable which is called ODBCINI and points to the odbc.ini file we just created:

```
[oracle@TARGET ggs]$ export ODBCINI=/home/oracle/ggs/odbc.ini
```

NOTE: POSTGRES SPECIAL

Security at Postgres may deny connections from other hosts, so check the Postgres config files:

Postgres conf file pg_hba.conf needs this config line:

```
        host    all             all             0.0.0.0/0           md5
```

so that ALL clients can connect. The sample above just means that all clients can connect to the Postgres database. Commonly this could show more restrictive setting depending on business rules, so it is always worth to have a look at this file when clients can not connect to the Postgres database using the ODBC driver.

A second config file is the Listener which is configured in the postgresql.conf. The parameter:

listen_addresses '*'

just means that Postgres listens on all available addresses. The settings here could be more restrictive and specify in the postgresql.conf for example :

```
listen_addresses = 'localhost'         # what IP address(es) to listen on;
```

so only localhost connections are possible.

We've prepared the ODBC set up, now let's start with the GoldenGate set up. Similar to the installation on the Oracle database host we first create the GoldenGate subdirectories:

```
[oracle@TARGET ggs]$ ./ggsci
GGSCI (TARGET.ORACLE.COM) 1> create subdirs

Creating subdirectories under current directory /home/oracle/ggs

Parameter files                /home/oracle/ggs/dirprm: already exists
```

```
Report files               /home/oracle/ggs/dirrpt: created
Checkpoint files           /home/oracle/ggs/dirchk: created
Process status files       /home/oracle/ggs/dirpcs: created
SQL script files           /home/oracle/ggs/dirsql: created
Database definitions files /home/oracle/ggs/dirdef: created
Extract data files         /home/oracle/ggs/dirdat: created
Temporary files            /home/oracle/ggs/dirtmp: created
Stdout files               /home/oracle/ggs/dirout: created
```

create the Manager parameter file and start the manager:

```
GGSCI (TARGET.ORACLE.COM) 2> edit param mgr
```

As my Postgres GoldenGate installation is on a different host than the GoldenGate installation for the Oracle database I can use again the 7809 port (if both set ups are on the machine, please make sure to choose a different port number):

```
PORT 7809
```

Once we created the parameter file we can start the manager and check its status:

```
GGSCI (TARGET.ORACLE.COM) 3> start mgr

Manager started.

GGSCI (TARGET.ORACLE.COM) 4> info all

Program     Status      Group       Lag at Chkpt  Time Since Chkpt

MANAGER     RUNNING
```

### 2. Demo table in Oracle and Postgres

As mentioned, it is a basic set up without using initial load nor a data pump process. So we only create a simple table in Oracle and Postgres to replicate data. My Oracle database has a user called postgres and this user will now own a table ggtest:

```
Oracle DB:

SQL> connect postgres/postgres
Connected.
SQL> create table ggtest (col1 number, col2 varchar2(20));

Table created.

SQL> alter table ggtest add primary key (col1);

Table altered.
```

Then you can connect with a Postgres utility to the Postgres database and create a similar table in the public schema:

```
PostgreSQL:

CREATE TABLE "public"."ggtest"
(
  "col1" integer NOT NULL,
  "col2" varchar(20),
```

```
  CONSTRAINT "PK_Col111" PRIMARY KEY ("col1")
)
```

As we don't have an extract nor replicat process it does not make sense to enter a record into the Oracle database yet. So let's continue with the set up and verify the database connections using GoldenGate.

### 3. Verify Oracle DB connection

To check the connection to the Oracle database we can use the GoldenGate command interface, log into the Oracle db, list the tables we can capture and check their data types:

```
GGSCI (SOURCE.ORACLE.COM) 8> dblogin userid postgres, password postgres
Successfully logged into database.


GGSCI (SOURCE.ORACLE.COM) 9> list tables *
POSTGRES.GGTEST

Found 1 tables matching list criteria.

GGSCI (SOURCE.ORACLE.COM) 10> capture tabledef POSTGRES.GGTEST
Table definitions for POSTGRES.GGTEST:
COL1                    NUMBER NOT NULL PK
COL2                    VARCHAR (20)
```

SIDE NOTE: The connection method I've chosen is based on a correct setting of the Oracle_SID environment variable. If your GoldenGate installation is on a different machine then the Oracle database you need to configure the SQL*Net. Details can be found in the manual.

### 4. Verify Postgres ODBC connection

To check the ODBC connection to the Postgres database we also use the GoldenGate command line tool, list the tables and check out the column definitions of the table we created in step 3:

```
GGSCI (TARGET.ORACLE.COM) 4> dblogin sourcedb gg_postgres userid postgres
Password:

2013-04-06 16:51:18  INFO    OGG-03036  Database character set identified as UTF-8. Locale: en_US.

2013-04-06 16:51:18  INFO    OGG-03037  Session character set identified as UTF-8.
Successfully logged into database.

GGSCI (TARGET.ORACLE.COM) 5> list tables *
public.ggtest

Found 1 tables matching list criteria.

GGSCI (TARGET.ORACLE.COM) 3> capture tabledef "public"."ggtest"
Table definitions for public.ggtest:
col1                                                                    NUMBER (10) NOT NULL PK
col2                                                                    VARCHAR (20)
```

So we can successfully connect to the Oracle database and to the Postgres database. Both connections are mandatory. Do not continue with the next steps unless both connections are working.

### 5. GoldenGate extract process

In the following section we create an extract process that captures the changes for the GGTEST table in the Oracle database and copies the changes directly to the Postgres machine. It's a simple set up to demonstrate the way a GoldenGate replication between Oracle and Postgres is working, but it should not be used without a data pump in production envs.

Every process needs it config file, so let's create it for the extract process

```
GGSCI (SOURCE.ORACLE.COM) 4> edit param epos
```

with these parameters:

```
EXTRACT epos
USERID postgres, PASSWORD postgres
RMTHOST TARGET.ORACLE.COM, MGRPORT 7809
RMTTRAIL ./dirdat/ep
TABLE postgres.ggtest;
```

My extract process is called epos and it connects as user POSTGRES using the password POSTGRES to the Oracle database. It will extract changes on the Oracle table ggtest stored in the postgres schema and will put the information into a trail file on my Postgres machine.

Once we created the parameter file we can add the extract process and start it:

```
GGSCI (SOURCE.ORACLE.COM) 5> add extract epos, tranlog, begin now
EXTRACT added.

GGSCI (SOURCE.ORACLE.COM) 6> add exttrail ./dirdat/ep, extract epos, megabytes 5
EXTTRAIL added.


GGSCI (SOURCE.ORACLE.COM) 7> start epos

Sending START request to MANAGER ...
EXTRACT EPOS starting



GGSCI (SOURCE.ORACLE.COM) 8> info all

Program     Status      Group       Lag at Chkpt  Time Since Chkpt

MANAGER     RUNNING
EXTRACT     RUNNING     EPOS        00:00:00      00:00:05
```

**6. Create DEFINITIONS File**

We're replicating data in a heterogeneous environment, so we need to give the process loading the data into the Postgres database more details about the data in the extract file. This is done by creating a definitions file using defgen. As usual we have to create a parameter file:

```
GGSCI (SOURCE.ORACLE.COM) 10> edit param defgen

DEFSFILE ./dirdef/GGTEST.def
USERID postgres, password postgres
TABLE POSTGRES.GGTEST;
```

Now exit from ggsci and call defgen on the command line and add the reference to the defgen parameter file just created:

```
[oracle@SOURCE ggs]$ ./defgen paramfile ./dirprm/defgen.prm

*****************************************************************************
        Oracle GoldenGate Table Definition Generator for Oracle
 Version 11.2.1.0.3 14400833 OGGCORE_11.2.1.0.3_PLATFORMS_120823.1258
   Linux, x64, 64bit (optimized), Oracle 11g on Aug 23 2012 16:58:29

Copyright (C) 1995, 2012, Oracle and/or its affiliates. All rights reserved.


                    Starting at 2013-04-06 16:48:09
*****************************************************************************

Operating System Version:
Linux
Version #1 SMP Wed May 26 10:38:10 EDT 2010, Release 2.6.18-194.3.1.0.2.el5
Node: SOURCE.ORACLE.COM
Machine: x86_64
                         soft limit   hard limit
Address Space Size   :    unlimited    unlimited
Heap Size            :    unlimited    unlimited
File Size            :    unlimited    unlimited
CPU Time             :    unlimited    unlimited

Process id: 16880

****************************************************************************
**            Running with the following parameters                       **
****************************************************************************
DEFSFILE ./dirdef/GGTEST.def
USERID postgres, password ********
TABLE POSTGRES.GGTEST;
Retrieving definition for POSTGRES.GGTEST



Definitions generated for 1 table in ./dirdef/GGTEST.def

Content of the Defgen File:

[oracle@SOURCE ggs]$ more ./dirdef/GGTEST.def
*+- Defgen version 2.0, Encoding UTF-8
*
* Definitions created/modified  2013-04-06 16:48
*
*  Field descriptions for each column entry:
*
*     1    Name
*     2    Data Type
*     3    External Length
*     4    Fetch Offset
*     5    Scale
*     6    Level
*     7    Null
*     8    Bump if Odd
*     9    Internal Length
*    10    Binary Length
*    11    Table Length
*    12    Most Significant DT
*    13    Least Significant DT
*    14    High Precision
*    15    Low Precision
*    16    Elementary Item
*    17    Occurs
```

```
*   18    Key Column
*   19    Sub Data Type
*
Database type: ORACLE
Character set ID: UTF-8
National character set ID: UTF-16
Locale: neutral
Case sensitivity: 14 14 14 14 14 14 14 14 14 14 14 11 14 14 14
*
Definition for table POSTGRES.GGTEST
Record length: 262
Syskey: 0
Columns: 2
COL1   64     50       0  0  0 1 0     50     50     50 0 0 0 0 1    0 1 2
COL2   64    200       56  0  0 1 0    200    200      0 0 0 0 0 1    0 0 0
End of definition
```

### 7. Copy DEFGEN File

As the replicat process needs details about the source database, we copy the generated definitions file located on our Oracle machine in the <GoldenGate home>/dirdef/GGTEST.def to the target machine where the Postgres database is installed into the <GoldenGate home>./dirdef/GGTEST.def directory.

### 8. Postgres replicat

Our extract process is set up to write all changes for the table GGTEST directly to the Postgres database machine into an extract file located in the dirdat directory. So we only need a process that reads those changes from the trail file and distributes it to the Postgres database. This process is called a replicat process running on the Postgres machine and it needs a parameter file

```
GGSCI (TARGET.ORACLE.COM) 1> edit param rpos
```

with the parameters:

```
REPLICAT rpos
SOURCEDEFS ./dirdef/GGTEST.def
SETENV ( PGCLIENTENCODING = "UTF8" )
SETENV (ODBCINI="/home/oracle/ggs/odbc.ini" )
SETENV (NLS_LANG="AMERICAN_AMERICA.AL32UTF8")
TARGETDB GG_Postgres, USERID postgres, PASSWORD postgres
DISCARDFILE ./dirrpt/diskg.dsc, purge
MAP POSTGRES.GGTEST, TARGET public.ggtest, COLMAP (COL1=col1,COL2=col2);
```

My replicat parameters are SOURCEDEFS which points to the definition file created in Step 6 and copied to the Postgres machine in step 7, two SETENV parameters where PGCLIENTENCODING is a Postgres parameter responsible for client encoding and commonly used when a client connects to the Postgres database and ODBCINI which refers again to the odbc.ini file created in step 1.

The TARGETDB parameter uses the ODBC DSN we created in the ODBC.INI file, the USERID and PASSWORD contain values for a valid Postgres user. for my "map" parameter I'm using a colmap setting as the case of the columns in my Postgres database are in small letters whereas Oracle by default puts everything into capital letters.

Create the replicat process, start it and verify if it is running:

```
GGSCI (ZKUPCHV119) 2> add replicat rpos,  NODBCHECKPOINT, exttrail ./dirdat/ep
REPLICAT added.
```

```
GGSCI (edvmr1p0) 3> start rpos

Sending START request to MANAGER ...
REPLICAT REPKG starting


GGSCI (ZKUPCHV119) 4> info all

GGSCI (TARGET.ORACLE.COM) 2> info all

Program     Status      Group       Lag at Chkpt  Time Since Chkpt

MANAGER     RUNNING
REPLICAT    RUNNING     RPOS        00:00:00      00:00:07
```

### 9. Testing the configuration

Finally we can now test the whole set up and insert a record into the Oracle database:

```
SQL> insert into ggtest values (1,'hello world!');

1 row created.

SQL> commit;

Commit complete.
```

and check with a POSTGRES tool if our newly inserted record is replicated:

```
-bash-3.2$ psql GGTest
psql (9.2.4)
Type "help" for help.

GGTest=# select * from ggtest;
 col1 |  col2
------+---------
   10 | hello world!

(1 rows)

GGTest=# \q
```

## REFERENCES

NOTE:1543702.1 - Character Conversion Issue When Replicating Data From Unicode Oracle To Unicode PostgreSQL

Didn't find what you are looking for?