## Architectures

One of the primary advantages with Oracle GoldenGate is the many architectures or use cases that this software can address. There are some common architectures that every organization implements depending on their requirements for replication. In this section, you take a look at the most common architectures that are used.

### Unidirectional Replication

The simplest and most common replication architecture for Oracle GoldenGate is the unidirectional replication architecture. This architecture is used to replicate data from a single source system to one or more remote target systems. Gaining a foundational understanding of this architecture will provide the basis that will be used as you start reviewing the more complex architectures that Oracle GoldenGate can provide. Figure [2-1](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_2_Chapter.html#Fig1) illustrates this architecture.



###### Figure 2-1. Unidirectional replication architecture

A unidirectional architecture consists of the core processes that are used in Oracle GoldenGate. These processes consist of the extract (capture) process, the data pump process, and the replicate (delivery) process. In between these processes are proprietary binary files called trail files. The trail files are used to store the captured transactions.

### Bidirectional (Active-Active) Replication

Building on the simplicity of the unidirectional replication architecture is the bidirectional (active-active) replication architecture. The bidirectional architecture is used to keep both the source and target systems in sync with data coming from both locations. Figure [2-2](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_2_Chapter.html#Fig2) illustrates how this architecture looks conceptually.



###### Figure 2-2. Bidirectional replication architecture

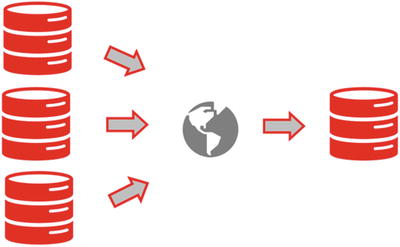
There are a few reasons for using a bidirectional architecture. The most common use case for a bidirectional architecture is during migrations when you are looking to upgrade your database systems. By using a pseudo-failback replication from the target, you can migrate your database and applications with a near-zero downtime approach. The pseudo-failback replication coming from the target side allows you to capture transactions and have an avenue to downgrade, if needed, without losing any transactions. The second use case for which people have been using bidirectional architecture for is a high-availability or disaster recovery solution. Although Oracle GoldenGate does not fall into Oracle’s disaster recovery category, many organizations have been using it this way due to the read/write functionality it allows. With the onset of cloud technologies, a bidirectional replication can be used to move data into and out of public or private cloud architectures.

###### Note

Do not confuse high availability with Oracle GoldenGate with disaster recovery technology such as Oracle Data Guard. These two technologies are similar but they are used for different purposes.

### Real-Time Data Warehousing

In today’s fast-paced business world, data warehousing is critical to many organizations. Data is a valuable commodity that is used by decision makers and affects every aspect of business. Oracle GoldenGate is positioned to be one of the best tools for consolidation where data is a concern. Oracle GoldenGate can be used to consolidate data from many heterogeneous platforms down to a single data warehouse. This capability effectively allows organizations to take data from any platform and perform data integration techniques and analysis from a single source of truth. Figure [2-3](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_2_Chapter.html#Fig3) provides a better conceptual view of this architecture.



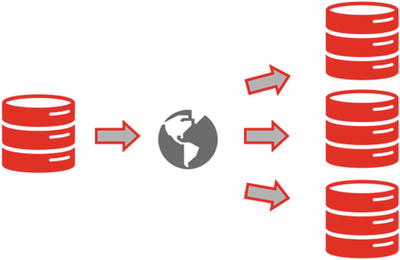
###### Figure 2-3. Consolidation replication architecture

With the ability to capture, transform, and apply data into a single source of truth database like a data warehouse, Oracle GoldenGate can be leveraged against other Oracle tools, allowing for a more robust integration process. Tools like Oracle Data Integrator can be leveraged alongside of Oracle GoldenGate to transform the data as the data is loaded into a data warehouse. Additionally, consolidation with Oracle GoldenGate can be extended to capture data from flat files, allowing a greater range of consolidation efforts from multiple sources.

As more and more organizations start to explore big data technology for decision making, Oracle GoldenGate can be used to consolidate data onto platforms like Hadoop. Using Oracle GoldenGate with big data will open up many more opportunities for organizations. At this point in the book, just be aware that Oracle GoldenGate can be used with current big data technologies.

### Real-Time Data Distribution

Oracle GoldenGate is a good tool for consolidating data from multiple sources to a single source of truth; it is just as good at distributing data to remote locations. In a data distribution architecture, Oracle GoldenGate takes data or subsets of data and ships that data to target databases that reside in different geographical regions. This approach allows for organizations to segregate data based on what is required for different lines of business or locations. Figure [2-4](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_2_Chapter.html#Fig4) illustrates how this architecture looks when implemented.

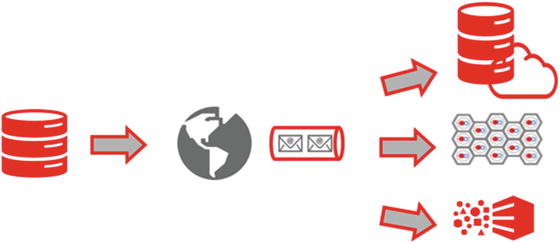


###### Figure 2-4. Data distribution architecture

By using a data distribution architecture, an organization can ensure that the correct geographical locations get the data that they require. This architecture also provides some sense of security for the data because the data being shipped is based on filters, within Oracle GoldenGate, that prevent data from being shipped to the wrong location. Overall, the data distribution architecture allows you to take a single source of truth and share the data based on predefined requirements for remote locations.

### Data Distribution via Messaging

Data distribution via messaging is similar to the data distribution architecture previously discussed; the only difference is in how data is actually shipped to the geographical locations. This architecture uses flat files for data distribution. Using flat files, Oracle GoldenGate can ensure that data can be captured and shipped to any type of system. Most systems that this architecture supports are various database types, office applications such as Microsoft Excel, and larger systems running on a big data framework. Figure [2-5](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_2_Chapter.html#Fig5) illustrates this architecture.



###### Figure 2-5. Data distribution via messaging architecture

As you can see, there are many different architectures in which Oracle GoldenGate can be used. The architectures highlighted in this chapter are just the tip of the iceberg. Oracle GoldenGate is such a flexible product that almost any type of architecture can be built for a wide range of use cases. This is the reason most organizations, after purchasing and exploring what it can do, start to find different business requirements that Oracle GoldenGate can address, leading to enterprise-wide adaptation of Oracle GoldenGate.

To understand where you can go with Oracle GoldenGate, you will need to take a closer look at what makes up these architectures. The simplest way of doing this is to use the unidirectional use case and dissect that to gain better understanding. Figure [2-6](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_2_Chapter.html#Fig6) shows the unidirectional use case in more detail.



###### Figure 2-6. Detail of the unidirectional architecture

You can see in Figure [2-6](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_2_Chapter.html#Fig6) is that there are a few different processes that are taking place. Starting from the left of Figure [2-6](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_2_Chapter.html#Fig6), you see the capture process, then trail files followed by a pump process (extract; more on this shortly). On the remote side, you see additional trail files feeding into a delivery process for applying to the target database. These processes make up the core of Oracle GoldenGate and allow for the product to be flexible and scalable no matter the environment. Let’s take a closer look at these processes.

## Understanding System Change Numbers

Now that you understand a few of the different architectures, getting an understanding of the system change number (SCN) is important. Oracle GoldenGate uses the SCN on an Oracle database to identify where the transactions are within the replication process. This number is also used when instantiating the replication process, so it is a very important piece of information to gather after setting up the capture process.

###### Note

The Oracle GoldenGate documentation uses the term change system number (CSN) in places. This means the same thing as SCN when using Oracle databases. Other databases like Microsoft SQL Server and MySQL will use a similar number. Consult your documentation for what you will need to use.

### Finding the System Change Number in Oracle

Within an Oracle database, finding the SCN is quite simple. This information can be found in the V$DATABASE view or by using the DBMS\_FLASHBACK package on the source database. Recording this number after starting the capture process will ensure that you have a consistent point to pull data from for instantiation purposes using Oracle Export Data Pump. Here’s are the queries you can use to pull this information:

SQL> select current\_scn from v$database;  
  
SQL> select dbms\_flashback.get\_system\_change\_number from dual;

Once you have the SCN, this number should be used with Oracle Export Data Pump to ensure you get a consistent read of the data that needs to be imported into the target system. Any transactions that are captured after the SCN will be placed in the trail files. All data that is exported while using the SCN will be handled by Oracle Export Data Pump. Once this data is imported to the target system along with transactions from the Oracle GoldenGate processes, you will have a complete set of data with a near-zero downtime approach on the target system.

###### Note

The GV$DATABASE view can also be used if using Oracle Real Application Cluster (RAC).

## GoldenGate Processes

What makes Oracle GoldenGate so flexible and scalable are the processes that are used to capture, transmit, and deliver data in a heterogeneous environment. These processes coupled with trail files for storing the transactions between processes can be configured to provide a robust environment. This section looks at each of the processes and how they are used within an Oracle GoldenGate environment.

### Manager Process

The manager process is not the most talked about process in the Oracle GoldenGate environment; however, it can be considered the brains of the process operation. The manager is responsible for many different parts within the environment. This is the one process that must be running, and remain running, on every system within a replication environment. Because it has to remain running on every system, the manager process is responsible for the following functions:

* Start and restart of Oracle GoldenGate processes
* Starting of dynamic processes
* Maintaining port numbers for processes
* Trail file management
* Event, error, and threshold reports

One manager process can control many different types of Oracle GoldenGate processes. This allows for central management of all things related to Oracle GoldenGate from a single home structure.

###### Note

If running in a Microsoft Windows environment, the manager process can be configured to run as a service.

### Collector Process

The collector process is a background process that runs on the delivery (target) side when online change synchronization is active. Collector processes are needed to ensure that the following tasks are performed:

* Connection requests from a remote extract to manager can be scanned and bound to available ports. Then assign the port number to the requesting extract via the manager process.
* Receive extracted transactions that are sent by the extract and write them to the trail file.

When a network connection is requested, the manager process automatically starts the collector process so there is no need for Oracle GoldenGate users to interact with the collector process. Because the collector can only receive information from only one extract, this means that there will be one collector started for each extract started. The collector process is terminated when the extract process is terminated.

### Capture Process

The extract process of Oracle GoldenGate is used to perform change data capture from the source database. The extract is used to synchronize data that is read from the online transaction log (in Oracle the online redo logs) or the associated archive logs. The data that is extracted, when configured for change synchronization, from the source database is then stored until it receives either a commit or rollback. On a commit, the extract persists the transaction to disk where it is stored in a series of files known as a trail file. Committed transactions will be stored in a trail file in sequentially organized transaction units. Once transactions are persisted to disk in a trail file, the transaction can be shipped using standard TCP/IP protocols.

###### Note

Trail files do not have to be shipped in all cases. Trail files can be consumed on the same local system or trail files can be shared via a shared file system such as Network File System (NFS).

There are two ways that an extract can be configured:

1. Initial load: The extract (captures) a current, static set of data directly from a source object. This is often referred to as a special run.
2. Change synchronization: The source data is synchronized with another set of data.

The initial load extract is used to perform static data set loads directly over the network. This process is normally considered a special run type of process. It is only intended to help get static data loaded into the target system and then stop running once that is complete. The change synchronization extract is your standard extract that allows you to synchronize data between the source and target. This is the most common type of extract that is used in most architectures. Both ways of using an extract are valid, but you will mostly likely use one over the other during normal operations.

Just as there are two ways of configuring the extract, there are two types of extract (capture) processes. The first is the classic capture, which is the basic version of Oracle GoldenGate capturing data. The second is the integrated capture. This version of the capture process takes the fundamental capture process from the operating system level and places it more in line with the database for replication purposes for Oracle databases. Both types of extract processes can be configured as either an initial load or change synchronization process. The following sections take a closer look at these two capture processes.

#### Classic Capture

The classic capture process is the default capture process for all Oracle GoldenGate setups. This type of capture process is controlled from the GGSCI utility but is closely managed from the operating system layer. This means that when there are problems, such as performance, the items needed to identify what is occurring have to come from the operating system level. Tools such as sar, mpstat, iostat, and free can be used to help diagnose what is occurring on the operating system and within Oracle GoldenGate.

The classic extract (capture) process also has a memory requirement that most people do not know of. Like any other process running on a machine, the process has to have a memory location from which to run. By default, the classic capture process will take between 25 and 55 MB of memory. This memory requirement ensures that data can be captured in an efficient way. Keep this in mind as you start building your Oracle GoldenGate environments.

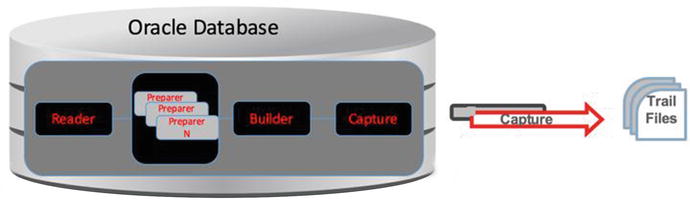
#### Integrated Capture

The integrated capture process is relatively new for Oracle GoldenGate. Integrated capture was introduced with GoldenGate release 11.2.0.2. At the time, integrated extract was only supported on Oracle Database versions 11.2.0.3 or later. Using an integrated capture requires the process to be registered with the Oracle Database. This allowed the capture process to interact with the log miner processes in the database.

###### Note

Integrated capture only works with Oracle Database versions 11.2.0.3 or later.

To really understand the integrated capture process, you need to take a closer look at the log miner processes inside of the Oracle Database. Figure [2-7](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_2_Chapter.html#Fig7) shows you the architecture for this type of process from a high level.



###### Figure 2-7. Integrated capture process

Looking at the image in Figure [2-7](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_2_Chapter.html#Fig7), you can tell there are a few internal database processes that are used to make the integrated process run. These processes are a part of the log miner configuration within the Oracle Database.

The log miner server is broken up into the following processes, which you will look at briefly:

* Reader: Used to read the online redo logs and split these files into sections to be scanned.
* Preparer(n): Used, in parallel if needed, to read the scanned regions of the online redo logs and prefilter the transactions.
* Build: Merges redo records identified by the preparer process and preserves them by SCN.
* Capture: Formats the redo records and places them into a logical change record (LCR), which places transactions into the local trail files.

As you can see, the integrated process is a bit more complicated than the classic process. Both processes have their benefits, although Oracle’s plan is to move toward the integrated process. Depending on the database platform and version, the classic process still has a place in the Oracle GoldenGate framework.

###### Note

The System Global Area (SGA) requires the   streams\_pool\_size to be set for better performance.

### Data Pump Process

The data pump group is a secondary extract group that is used to help ship data across network. Although a data pump is another extract group similar to the capture process, don’t confuse the two. The main purpose of the data pump process is to ship trail files across the network to the remote target system.

###### Note

Best practice is to rename the trail file when it reaches the target system.

If the data pump, if shipping trail files, is not configured, the primary extract group will write directly to the remote trail file.

With the data pump process being mostly used for shipping trail files across the network, why would you want to use a data pump process? The single, largest advantage of using a data pump process within your Oracle GoldenGate architecture is that a data pump helps protect against network failures. This ability to protect the captured transactions from being lost is huge for many business reasons. If your network were to have an outage, there are two scenarios that can occur:

1. With a network outage, the data pump process will continue to collect the trail files generated by the primary capture process and store them until the network is restored.
2. If you are not using local trail files and only writing remote trail files, i the case of a network outage, the primary extract will fail. Once the network is restored, the primary extract can be restored with no loss of transactions.

These scenarios highlight why many organizations need to ensure that network discussions are a part of the planning sessions for Oracle GoldenGate.

Unlike the capture process, the data pump does not have more than one configuration that you can use; classic mode is the only configuration available. The associated footprint that the data pump takes is much smaller than the classic capture process, but it still requires memory to run. Remember in most cases, the data pump process is used to ship trail files across the network.

### Delivery Process

The delivery process is the apply process within an Oracle GoldenGate environment. The delivery process is responsible for reading the trail files and applying the transactions found in chronological order. This ensures that the data is applied in the same order it was captured (SCN order). Until recently, there was only one version of the delivery process, the classic replicate. Starting in Oracle GoldenGate 12c (12.1.2.0), there are now three distinct versions of the replicate:

* Classic
* Coordinated
* Integrated

Each of these modes provides some sort of benefit, depending on the database to which it is being applied. Oracle is pushing everyone to a more integrated approach; however, you have to be on database version 11.2.0.4 at a minimum.

#### Classic Delivery

The classic delivery process is the default delivery process for basic Oracle GoldenGate configurations. This type of delivery process is managed from the operating system layer. Just like the classic capture process, this means when there are problems, such as performance, the items needed to identify what is occurring have to come from the operating system level.

This version of the delivery process also has a memory requirement that mirrors the classic capture process. Like any other process running on a machine, it has to have a memory location to run. By default, the classic delivery process will take between 25 and 55 MB of memory. This memory requirement is needed to sure that data is organized and processed in an efficient way.

#### Coordinated Delivery

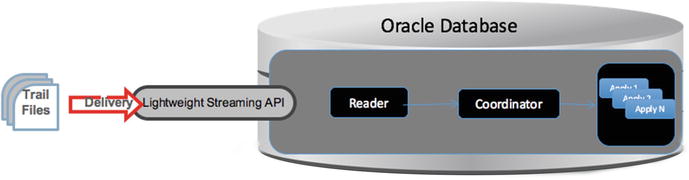
The coordinated delivery process is similar to the classic delivery process. The difference here is that the coordinated delivery process will spin up slave processes that are coordinated by a master delivery process. The purpose of the coordinated delivery process is to help split up the delivery process. The parallel process the coordinated delivery uses can be applied to large transactions as well as small transactions.

#### Integrated Delivery

The integrated delivery process has been introduced with the release of Oracle GoldenGate 12c. This type of delivery brings the ability for an Oracle database to ingest large amounts of transactions without dependence on specialized functions to help split the data into manageable groupings. This approach tracks the dependencies between transactions and applies the transactions based on primary key, foreign key, or unique key constraints, as they are processed.

Just like the integrated capture process, to get a better understanding of how this delivery process works, you need to take a closer look at the log miner architecture (Figure [2-8](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_2_Chapter.html#Fig8)). Within the database, the following processes are used to ingest the transactions that are coming from the trail files:

* Receiver
* Preparer
* Coordinator
* Apply(n)



###### Figure 2-8. Integrated delivery process

Each of one of these internal processes has a specific function for applying the transactions to the database.

* Reader: This is the inbound server that computes the dependencies among the transactions in the trail files based on constraints defined at the target database (PK, UK, FK). Data Definition Language (DDL) and barrier transactions are maintained and managed automatically.
* Coordinator: This ensures that all records are maintained in the order of transaction and then passes the record on to the apply processes.
* Apply(n): These are the processes that apply the records to the database in order of SCN; however, transactions that do not have interdependencies can be safely executed and committed out of order to achieve faster throughput. Parallelism of the apply processes is determined by the amount of records and the load on the receiving database.

###### Note

Just like the integrated capture process, the streams\_pool\_size needs to be configured to ensure proper performance.

## Trail Files

In many of the architectural images in this chapter, there is an image labeled “trail files .” What exactly are trail files? They are Oracle GoldenGate-specific binary files used to ship transactions within the Oracle GoldenGate architecture. These files support the continuous extraction and replication of database changes by storing records of captured changes temporarily on disk. Trail files can exist on a source system, known as local trail files. They can also be stored on the target system, known as remote trail files. By using trail files as storage, Oracle GoldenGate supports data accuracy and fault tolerance. The usage of trail files also allows you to keep Oracle GoldenGate processes independent of each other. This gives you greater flexibility within your Oracle GoldenGate framework and more control over how data is processed and delivered.

## Profiler

In preparing an Oracle database for replication with Oracle GoldenGate, you will need to identify anything within the database that might not be able to be replicated. For this purpose, Oracle has created an Oracle GoldenGate Profiler . The profiler is intended to query the database of all the nondefault users to identify the current configuration and any unsupported data types or data types that might need special attention. The latest profiler script can be located on My Oracle Support.

###### Note

My Oracle Support Notes 1298562.1 and 1296168.1 providethe scripts for all schemas or single-schema profiling.

### Run Profiler

Once you have downloaded the profiler from My Oracle Support (MOS) , running it is quite simple. All you need is access to SQL\*Plus and SYSDBA access. Listing [3-1](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1) provides a glimpse at how the script is run.

###### Listing 3-1. Running Profiler

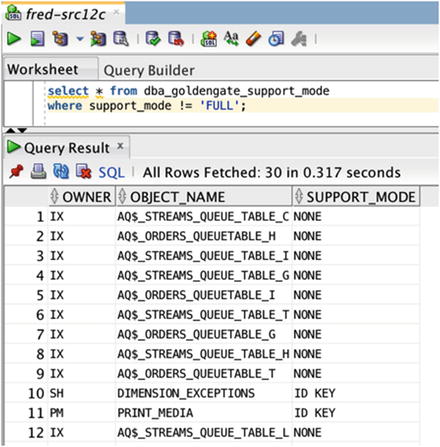
$ sqlplus / as sysdba  
SQ> @full-DB\_CheckOracle\_07082015sql

After the profiler has gathered all the information it requires from the database, an output file will be produced with the results. Review the output file and verify there is nothing of concern that you need to address before setting up replication. Listing [3-2](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1) provides an example of what the profiler report looks like for an older version of the Oracle database.

###### Listing 3-2. Profiler Output

------ System Info:  
  
DateTime:  
-------------------  
10-24-2014 12:50:02  
  
BANNER  
----------------------------------------------------------------  
Oracle Database 10g Enterprise Edition Release 10.2.0.5.0 - 64bi  
PL/SQL Release 10.2.0.5.0 - Production  
CORE   10.2.0.5.0   Production  
TNS for IBM/AIX RISC System/6000: Version 10.2.0.5.0 - Productio  
NLSRTL Version 10.2.0.5.0 - Production  
  
NAME      LogMode      SupLog:  PK  UI  For FK  All Created  
--------- ------------ -------- --- --- --- --- --- -------------------  
TEST1234  ARCHIVELOG   NO       NO  NO  NO  NO  NO  10-03-2010 15:45:32  
  
PLATFORM\_NAME  
--------------------------------------------------------------------------------------------  
AIX-Based Systems (64-bit)  
  
------ Objects stored in Tablespaces with Compression are not supported in the current release of OGG  
  
------ Distinct Object Types and their Count By Schema:  
  
OWNER           OBJECT\_TYPE               TOTAL  
--------------- -------------------- ----------  
TSMSYS          TABLE                         1  
TSMSYS          INDEX                         1  
ORACLE\_OCM      JOB                           2  
EDS             TABLE                       158  
VPDADM          FUNCTION                      5  
EDS             FUNCTION                      2  
EDS             TRIGGER                      40  
EPHARM          TRIGGER                       2  
FDB\_2           TABLE                       244  
EAS             TABLE                        12  
CS\_CLINICAL     TABLE                        22  
SBMO            TYPE                          1  
MDS             SEQUENCE                      1  
DIB3\_AUX        TABLE                         8  
VPDADM          PROCEDURE                     2  
MDS             PROCEDURE                     1  
WL\_CPR\_REGLOG   SYNONYM                       6  
REGISTRATION\_LO PACKAGE                       1

Starting with Oracle Database 11g (11.2.0.4) and later versions, the Oracle GoldenGate Profiler script has been replaced with a database view, DBA\_GOLDENGATE\_SUPPORT\_MODE. From this database view, you can query what schemas and objects will not be supported with Oracle GoldenGate. Figure [3-1](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1) shows that there are schemas and tables that are not supported.



###### Figure 3-1. Output from DBA\_GOLDENGATE\_SUPPORT\_MODEs

Notice that Owner IX has a few objects that are not supported by Oracle GoldenGate, indicated by the NONE value in the SUPPORT\_MODE column. If all columns in the table were supported by Oracle GoldenGate, the SUPPORT\_MODE column would indicate this with a value of FULL. When you see a value of ID KEY, this means that the identity column of the object can be used in replication configurations.

At this point, you should understand how to confirm if your database, schemas, or both can be replicated for Oracle GoldenGate. For database versions that are older than 11.2.0.4, you should use the profiler scripts that Oracle provides. If databases are version 11.2.0.4 or newer, you should use the DBA\_GOLDENGATE\_SUPPORT\_MODE view to look at what is supported.

Additionally, although you have focused on Oracle databases in this chapter, the profiler scripts also support the heterogeneous platforms that Oracle GoldenGate supports. These profiler scripts can be found in My Oracle Support using the MOS notes provided earlier.

## Parameter Files

Part of the configuration process in setting up Oracle GoldenGate is establishing the parameter files for each of the processes within the environment. The parameter files are simple text files that can be edited with any text editor. These files can also be precreated before creating the required processes within Oracle GoldenGate from the GGSCI.

Parameter files are the runtime brains within the Oracle GoldenGate environment. The settings that are placed in the parameter files tell Oracle GoldenGate what to capture from the source, where to ship the data, and how to apply the data to the target side. In small environments, you might have as few as two parameter files to worry about. In larger environments, the number of parameter files can increase dramatically depending on the needs of the environment and architecture.

### Extract Parameter File

Let’s take a look at a simple parameter file that can be used to capture data from an Oracle database. Listing [3-3](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1) provides a view into what a parameter file might look like for a capture process.

###### Listing 3-3. Simple Extract Parameter File

EXTRACT E\_HR  
USERID ggate, PASSWORD ggate  
SETENV (ORACLE\_HOME="/u01/app/oracle/product/11.2.0/db\_3")  
SETENV (ORACLE\_SID="tst12c")  
TRANLOGOPTIONS DBLOGREADER  
 EXTTRAIL ./dirdat/bt  
TABLE HR.\*;

The way that Oracle GoldenGate processes parameter files is by taking a top-down approach. If you take the same approach, you can understand what the process is doing on startup and replication. In Listing [3-3](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1), you can see that this is for a capture process named E\_HR, uses a user named ggate to access the database, and sets up the environment to access the tst12c database. After the environment is set, the capture process is looking to read from the transaction log. TRANLOGOPTIONS is the parameter that controls the way the extract will interact with the database transaction log. By using the DBLOGREADER option, the extract knows to use a new application programming interface (API) that is available in Oracle 11.2.0.2 and later. This API uses the database server access to mine the redo and archive logs. After accessing the transaction log, the extract checks for any long-running transactions on a 30-minute interval, and moves captured data to the defined local trail file. Finally, the capture process captures all Data Manipulation Language (DML) coming from tables in the HR schema.

###### Note

The CHECKPARAMS parameter can be used for all processes. This parameter allows you to check the syntax of the parameter file before replication starts. Although CHECKPARAMS is available to use with the parameter file, starting in Oracle GoldenGate 12c (12.2), there is a new checkprm utility.

### Data Pump Parameter File

Just like the extract parameter file, the data pump parameter file provides similar information. The distinct difference here is that the data pump process is used to ship the trail file and captured transactions to a remote location on the network. Listing [3-4](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1) provides a view of how simple the data pump parameter file can be.

###### Listing 3-4. Data Pump Parameter File

-- Verifies parameter file syntax. COMMENT OUT AFTER TESTING.  
CHECKPARAMS  
EXTRACT P\_HR  
PASSTHRU  
RMTHOST 172.16.15.132, MGRPORT 15000, COMPRESS  
RMTTRAIL ./dirdat/rt  
TABLE HR.\*;

###### Note

Starting in Oracle GoldenGate 12c (12.2), the PASSTHRU parameter is deprecated due to metadata being shipped in the trail files.

When reading this parameter file, you can tell that an extract process is set up as a data pump when you see the parameter PASSTHRU. If the PASSTHRU parameter is not included, then the extract will operate as a normal extract. Additionally, in the parameter file you will see where the trail files and transactions are shipped to with the remote host (RMTHOST) parameters. As part of the RMTHOST parameter, you need to tell the data pump how to connect to the remote server with the MGRPORT option with port number. Finally, the COMPRESS option forces the data packets to be compressed across the network. The RMTTRAIL parameter tells Oracle GoldenGate to create a new trail file on the remote side, containing data and transformations, when they are shipped. The data pump needs to know what data is being shipped.

###### Note

The MGRPORT is the port of the manager process on the target side. The number can be different for every manager.

### Replicat Parameter File

The replicat parameter file is the parameter file that tells Oracle GoldenGate what to apply. It provides the mapping command that links the incoming transactions to the targeted tables. In active-active or multimaster environments, this file is where you would configure conflict detection and resolution (CDR) to help resolve conflicts as they occur. Listing [3-5](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1) displays a simple apply file for a replicat process .

###### Listing 3-5. Replicat Parameter File

REPLICAT R\_HR  
SETENV (ORACLE\_HOME="/u01/app/oracle/product/11.2.0/db\_3")  
SETENV (ORACLE\_SID="tst12cr")  
USERID ggate, PASSWORD ggate  
map HR.\*, target HR.\*;

Just like the extract parameter file, the replicat parameter file needs to set up the environment that is needed to connect to an Oracle database. You then need to provide a username and password to interact with the database. In older versions of Oracle GoldenGate, the parameter ASSUMETARGETDEFS is used to correlate the metadata structure between source and target if the tables match. If the metadata of tables are different, then this parameter needs to be replaced with a SOURCEDEF file that supports the mapping. At the end of the file, you see the map statement. This statement maps the incoming transactions to the target side.

### Manager Parameter File

The manager process is the process that keeps track of all the other processes in the Oracle GoldenGate environment. Out of all the processes, this parameter file is the simplest to configure. Listing [3-6](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1) shows how simple it is.

###### Note

Notice you do not need to name the manager. In Oracle GoldenGate environments, the manager process has a default name of MGR, so the parameter file has to be named MGR.prm.

###### Listing 3-6. Manager Parameter File

PORT 15000

Listing [3-6](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1) shows a simple parameter file for the manager process. The only thing required in the manager parameter file is the port number that the manager will listen on. This allows Oracle GoldenGate instances to connect with each other and write trails files. Additionally, as mentioned earlier, the manager process is used to keep track of items like events, up or down status of processes, and errors that occur. To make better use of the items that the manager keeps track of, the parameter file can be expanded on. Listing [3-7](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1) shows a detailed version of the manager parameter file with additional details.

###### Listing 3-7. Detailed Manager Parameter File

PORT 15000  
DYNAMICPORTLIST 15010-15035  
PURGEOLDEXTRACTS ./dirdat/\*, USECHECKPOINTS, MINKEEPDAYS 2  
AUTORESTART ER \*, RETRIES 6, WAITMINUTES 2, RESETMINUTES 30  
LAGCRITICALSECONDS 30  
LAGREPORTMINUTES

In this version of the manager parameter file, you see there are more parameters added to tell the manager what to do with old trail files (PURGEOLDEXTRACTS), when to restart (AUTORESTART) the other processes if they are terminated abnormally, and when to check for lag (LAGCRITITCALSECONDS/LAGREPORTMINUTES). You will also notice there is a parameter to use dynamic ports. The DYNAMICPORTLIST parameter is used to help Oracle GoldenGate communicate beyond firewalls if needed and limits port allocations to specific ranges for communication. This is helpful, because Oracle GoldenGate will grab random unused ports otherwise.

## Trail Files

Chapter [2](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_2_Chapter.html) touched on the concept of trail files, the files that Oracle GoldenGate uses to store committed transactions that are captured. On the capture side of the environment, the trail file is considered the local trail file because it is on the capture side of the setup. The local trail file is created by the capture process and then referenced by the data pump process for shipping. As the file is shipped, the data pump process creates the remote trail file at the target location. The apply process will then read the remote trail file to process the transactions on the target database.

### Sizing and Retention of Trail Files

Sizing of the trail files is always a question that comes up when building an Oracle GoldenGate environment. By default, the size of a trail file will be 50 MB. For many environments, this size is too small, so how do you effectively size the trail files for your environment? Once they are sized, how do you retain them in the case of disasters?

Here are a few recommendations that can be followed:

1. Make the size the same as archive logs.
2. Identify the size and frequency of redo log switches and use that as a guide.
3. Monitor lag and trail file switch, based on size, and readjust as needed.
4. Retention of trail files should be handled similar to retention of archive logs.

To address the first two recommendations, the best way to look at this is to use the same rules that you would apply when looking at the database. If the database archive logs are configured at intervals to switch up to four times per hour, then your trail files should be sized similarly. By keeping the archive log switching and the trail file switching the same, you will ensure proper retention of the trail files along with cutting down on network bandwidth.

As for the third recommendation, your trail files should be on average between 10 percent and 30 percent of the archive log sizes if you are replicating everything. This value should be smaller if you are only doing a few schemas or less.

After you have sized the trail files, you should be following the same process for retaining them as you would use for your database. Prior to Oracle GoldenGate 12c (12.2), retaining trail files was critical, especially on the target side. With the release of Oracle GoldenGate 12c (12.2), the data pump process keeps track of the trail files that are shipped and created on the target side based on the checkpoints. If there is a need to recover the remote trail files, you just need to clear the trail files from the target side and then stop and restart the data pump process. This will re-create the remote trail files that you need for processing.

## Obey Files

Obey files are files that you create to run against an Oracle GoldenGate environment. The content of an obey file is a sequenced set of Oracle GoldenGate commands that can be run from a single file. Listing [3-8](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1) provides an example of an obey file that is used to create a capture and data pump process.

###### Listing 3-8. Obey File Example

--Adds Extract process  
ADD EXTRACT E\_HR, TRANLOG, BEGIN NOW  
  
--Adds local trail file  
ADD EXTTRAIL ./dirdat/lt, EXTRACT E\_HR, megabytes 1000  
  
--Adds Data Pump Process  
ADD EXTRACT P\_HR, EXTTRAILSOURCE ./dirdat/lt  
  
--Adds remote trail file  
ADD RMTTRAIL ./dirdat/rt, EXTRACT P\_HR, megabytes 1000

You will notice in Listing [3-8](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1) that all the commands have to do with adding the processes to Oracle GoldenGate. Obey files are created for setting up environments and they can be created for any command you would like to run within the Oracle GoldenGate environment.

Until now, you have taken a look at the profiler that is run against an Oracle database to see if there is anything you need to worry about before setting up your replication environment. You have also taken a look at the different parameter files that are needed for the components of the replication environment. Now, let’s take a look at how to put these pieces together in a simple replication environment.

## Unidirectional Configuration

The simplest configuration that can be done with Oracle GoldenGate is the unidirectional or one-way replication. This architecture is used for moving data from one database to another, mostly in migrations or reporting architectures with Oracle GoldenGate. To set up unidirectional replication, there are a series of steps that have to be completed. These steps are broken down into three distinct categories:

1. Prerequisites.
2. Instantiation.
3. Apply.

The largest of these categories is the prerequisites category. For many Oracle GoldenGate architectures, this is where you will spend a bulk of your time until replication is started. By spending your time on the prerequisites, you will ensure that the capture process is done correctly and captures all the required data. This will make the overall replication process simpler and cleaner.

### Prerequisites

As mentioned previously, the prerequisites are the part of the Oracle GoldenGate configuration on which you will spend a good bit of time. Some of the steps in the prerequisites will also be done on the target (apply) side of the configuration as well. The prerequisites cover several areas of database configuration for replication, including these:

1. Archive Log mode
2. Logging modes (supplemental logging/force logging)
3. GoldenGate Parameter
4. GoldenGate Users
5. GoldenGate Extract Parameters
6. Trandata (table supplemental logging)
7. Building Extract (Capture)
8. Building Data Pump (Shipping)

Following these prerequisites, you should be able to have your unidirectional replication up and running in no time.

#### Enable Archive Log Mode

With any Oracle database, the recoverability of the database is based on the data that can be recovered. Oracle provides for recoverability of the database using the online redo logs and archive logs. By default, Oracle uses online redo logs in No Archive Log mode to provide a minimal form of recoverability for transactions during rollback operations. Although this is good, it is not valid for Oracle GoldenGate operations against an Oracle database. An Oracle database needs to be placed into Archive Log mode for recoverability purposes and archiving purposes.

###### Note

For more information on archive log items, refer to the Oracle documentation at <http://docs.oracle.com/database/121/ADMIN/archredo.htm#ADMIN11332>

To enable Archive Log mode within the Oracle Database, follow these steps:

1. Shut down the database.

$ sqlplus / as sysdba  
SQL> shutdown immediate

1. Mount the database.

SQL> startup mount;

1. Alter the database.

SQL> alter database archivelog;

1. Open the database.

SQL> alter database open;

1. Verify that the database is in Archive Log mode.

SQL> archive log list;

#### Enable Supplemental Logging

Within the Oracle database framework, the redo log files are used for instance recovery and media recovery when there is a failure. Data that are needed for recovery are automatically recorded to the redo logs; however, supplemental logging expands on the logging of columns within a table. By default on an Oracle database installation, supplemental logging is not enabled, meaning that if a column is not changed, it is not going to make it into the redo.

###### Note

Supplemental logging, at a minimum, must be enabled before the log files can be used with LogMiner.

By turning on supplemental logging at the database level, you are ensuring that the database is logging the minimal amount of data needed for recovery. Oracle GoldenGate leverages this supplemental logging to ensure that transactions can be captured and used. Later in this chapter, we look at how Oracle GoldenGate allows you to enable supplemental logging at the table level using schematrandata or trandata. For now, know that supplemental logging at the database level has been enabled for use with Oracle GoldenGate. To turn on supplemental logging, the SQL statement in Listing [3-9](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1) should be run against the database.

###### Listing 3-9. Enable Supplemental Logging

SQL> alter database add supplemental log data;

After enabling supplemental logging within the database, verify that it was enabled by looking at the SUPPLEMENTAL\_LOG\_DATA\_MIN column in the V$DATABASE view using the SQL in Listing [3-10](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1).

###### Listing 3-10. SQL to Verify Supplemental Logging

SQL> select supplemental\_log\_data\_min from v$database;

The output from SQL to check if supplemental logging is enabled will return a YES value if it is enabled (Listing [3-11](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1)).

###### Listing 3-11. SQL Output for Supplemental Logging

SQL> col supplemental\_log\_data\_min format a30;  
SQL> select supplemental\_log\_data\_min from v$database;  
SUPPLEMENTAL\_LOG\_DATA\_MIN  
------------------------------  
YES

Once supplemental logging is enabled, the minimal amount of columns needed to track the change will be captured and recorded to the redo logs.

#### Enable Force Logging

Unlike supplemental logging, force logging is used to ensure all changes in the database except changes to temporary tablespaces and temporary segments are captured. Force logging does exactly what is refers to: It forces the capture of changes within the database. This setting takes precedence over and is independent of any logging options set for individual tablespaces and individual database objects.

For Oracle GoldenGate, force logging is good because it ensures that all changes are captured regardless of tablespace or object that needs to be captured. Coupled with supplemental logging, all SQL transactions performed against the database will be captured and recorded to the redo logs.

###### Note

Force logging mode can have performance effects on the database.

To enable force logging at the database level, use the SQL shown in Listing [3-12](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1).

###### Listing 3-12. Enable Force Logging

SQL> alter database force logging;

After force logging has been enabled, it can be checked using the FORCE\_LOGGING column of the V$DATABASE view (Listing [3-13](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1)).

###### Listing 3-13. Checking for Force Logging

SQL> select force\_logging from v$database;  
  
                FORCE\_LOGGING  
                ---------------------------------------  
                YES

With force logging enabled, the database is now configured to ensure that all transaction types are captured to the redo logs. This will be critical to ensuring that all transactions are captured by Oracle GoldenGate.

#### Enable GoldenGate Parameter

Starting with Oracle Database 11g (11.2.0.4 or later), to use Oracle GoldenGate a new parameter needs to be enabled. This parameter is a Boolean parameter that tells the database that Oracle GoldenGate is going to be used to perform replication within the database framework.

To show the default settings of this parameter, issue a show parameter command from the SQL plus prompt as illustrated in Listing [3-14](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1).

###### Listing 3-14. Checking for GoldenGate Parameter

SQL> show parameter goldengate  
  
NAME                                 TYPE        VALUE  
------------------------------------ ----------- ----------------  
enable\_goldengate\_replication        boolean     FALSE

Notice that the default setting for the parameter is set to FALSE. To change this setting, an ALTER SYSTEM command needs to be issued. Changing this parameter can be done dynamically and requires no bouncing of the database, as illustrated in Listing [3-15](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1).

###### Listing 3-15. Enable the Database to Use Oracle GoldenGate Replication

SQL> alter system set enable\_goldengate\_replication=true scope=both;

After enabling the database, it can be verified by using the SHOW PARAMETER command (Listing [3-16](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1)):

###### Listing 3-16. Verifying GoldenGate Parameter

SQL> show parameter goldengate;  
  
NAME                                          TYPE           VALUE  
------------------------------------ ----------- ----------------------  
enable\_goldengate\_replication        boolean    TRUE

The next step in the prerequisites is to create the user that will be used to manage the Oracle GoldenGate environment. This user is a standard Oracle database user and will need to be created on both the source and target databases.

### Create a GoldenGate User

Before building out the GoldenGate environment for a unidirectional implementation, you first have to have a database user that will be referred to as the GoldenGate user. This user will be used for Oracle GoldenGate to interact with the database. This user is a standard database user with a few special permissions to ensure that replication can be performed.

The simplest way to create a user is to use the defaults that the Oracle database provides. An example of creating the GoldenGate user is provided in Listing [3-17](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1).

###### Listing 3-17. Creating the GoldenGate User

create user GGATE  
identified by ggate  
default tablespace USERS  
temporary tablespace TEMP  
quota unlimited on USERS  
account unlock  
/

As you can tell, creating the GoldenGate user is quite simple. In any GoldenGate configuration, this user needs to be created on both sides of the replication environment to ensure that GoldenGate users are easily identified and controlled between environments.

### Granting Permissions to GoldenGate Users

GoldenGate users require a set of permissions to ensure that replication activities can perform for any updates, inserts, deletes, and Data Definition Lanuague (DDL) changes. These permissions are granted by using grant and revoke statements as needed with the Oracle ecosystem. Table [3-1](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Tab1) illustrates the permissions need for the GoldenGate user.

###### Table 3-1. Permissions for Oracle GoldenGate Users

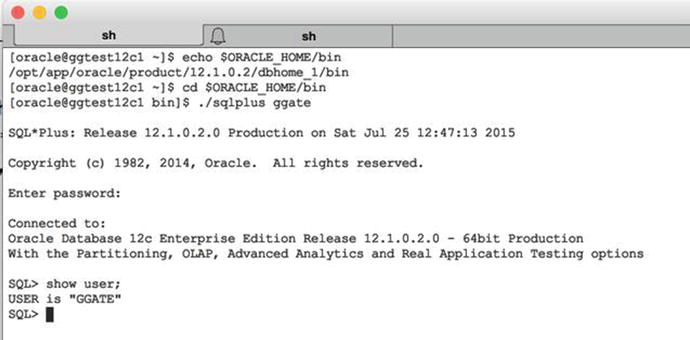
| **Permission** | **Purpose** |
| --- | --- |
| Create session | Allows GoldenGate user to create a session |
| Resource | Allows GoldenGate user to create database objects |
| Connect | Allows GoldenGate user to connect to database |
| Select any dictionary | Allows GoldenGate user to view the data dictionary |
| Flashback any table | Allows GoldenGate user to perform flashback operations on a table |
| Select any table | Allows GoldenGate user to select from any table |
| Select on dba\_clusters | Allows GoldenGate user to read DBA\_CLUSTERS |
| Execute on dbms\_flashback | Grants GoldenGate user permission to flashback procedures |
| Select any transaction | Allows GoldenGate user to select any transaction |
| Lock any table | Allows GoldenGate user to lock any table |
| Insert any table | Allows GoldenGate user to insert any table |
| Update any table | Allows GoldenGate user to update any table |
| Delete any table | Allows GoldenGate user to delete any table |
| Create table | Allows GoldenGate user to create tables in their own schema |

If you are building your GoldenGate user on 11.2.0.4 or later databases, you have the option to use the DBMS\_GOLDENGATE\_AUTH.GRANT\_ADMIN\_PRIVILEGE procedure. This procedure grants all the same permissions that are discussed in Table [3-1](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Tab1).

###### Note

Many of these privileges are also covered in the DBMS\_STREAMS\_AUTH package; however, it appears that they don’t always take through the package so granting them directly through SQL might be easier.

Once the GoldenGate user is created and privileges are granted, this is all that is needed within the database at this time. Verify that the user works by logging in from SQL\*Plus or some other SQL utility. Figure [3-2](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig2) demonstrates that logging in with the GoldenGate user was successful.



###### Figure 3-2. Logging in as GGATE

After validating that the GoldenGate user works, it is time to move on to setting up the Oracle GoldenGate processes.

### Building Source GoldenGate Environment

Now that the database is configured to be used with Oracle GoldenGate, the next step is to build the source side of the replication environment. As discussed earlier, you will need a parameter file for each of the processes on the source side. These processes consist of a manager process, an extract (capture) process, and the data pump process.

###### Note

The parameter files can be created ahead of time with any text-based editor.

To begin working on the parameter files, you need to access the Oracle GoldenGate environment. To do this, navigate to the Oracle GoldenGate home directory where Oracle GoldenGate is installed.

###### Note

To make this easier, setting the Oracle GoldenGate information in /etc/oratab will enable quick setup for accessing the Oracle GoldenGate binaries.

To access the binaries, change directories to $OGG\_HOME. As you navigate to the binaries also make sure that Oracle Home ($ORACLE\_HOME) is set to the database binaries needed to support replication. See Listing [3-18](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1).

###### Listing 3-18. Navigating to GoldenGate Home

[oracle@ggtest12c1 ∼]$ env | grep \_HOME  
OGG\_HOME=/opt/app/oracle/product/12.1.2/oggcore\_1  
ORACLE\_HOME=/opt/app/oracle/product/12.1.0.2/dbhome\_1  
[oracle@ggtest12c1 ∼]$ cd $OGG\_HOME  
[oracle@ggtest12c1 oggcore\_1]$

###### Note

If working from the command line a lot, it might be a good option to look into rlwrap. This tool allows you to store a command history that can be toggled through using the arrow keys on the keyboard. See [http://utopia.knoware.nl/~hlub/uck/rlwrap/](http://utopia.knoware.nl/%7Ehlub/uck/rlwrap/) .

After accessing the Oracle Home for GoldenGate, call the GGSCI utility to take you into the GoldenGate environment. Before you build any GoldenGate processes, it is a good idea to create the table-level supplmental log groups needed for GoldenGate by running either schematrandata or trandata once you log in to the database using the GoldenGate user you configured. The next section looks at how to add these options.

### Adding SCHEMATRANDATA or TRANDATA

There are two types of trandata that can be used within Oracle GoldenGate. Both types do the same thing, but one is more specific to the schema level and the other is more specific to the database level. Both approaches can be used with enabling trandata, though. This section explains what each type of trandata is used for and why you should understand the use of it.

#### SCHEMATRANDATA

When you use SCHEMATRANDATAto enable schema-level supplemental logging for tables, logging will act on all of the current and future tables in an associated schema. This option automatically logs a superset of available keys that Oracle GoldenGate requires for row identification. This option is valid for both classic and integrated capture modes with Oracle GoldenGate.

ADD SCHEMATRANDATA provides the following benefits:

* Enables Oracle supplemental logging for new tables created with the CREATE TABLE command.
* Updates supplemental logging information for tables affected by an ALTER TABLE to add or drop columns.
* Updates supplemental logging for tables that are renamed.
* Updates supplemental logging for tables where unique or primary keys are added or dropped.

An additional benefit of using ADD SCHEMATRANDATA is that by default it logs the key columns of a table in the following order of priority:

1. Primary keys.
2. If there are no primary keys, all unique indexes will be used for the table, including those that are disabled, unusable, or invisible. Additionally, foreign keys will be used for row dependency.
3. If the prior two are not available, the all scalar columns of the table will be logged.

#### TRANDATA

When you have a need to capture the transaction information from the transaction records, then it is time to use ADD TRANDATA. ADD TRANDATAis valid for specific databases like, IBM DB2, DB2 LUW, DB2 z/OS, MS SQL Server, and a few others, including the Oracle Database. This option expands on the information captured during the extraction process. Oracle recommends that force logging be enabled along with minimal supplemental logging at the database level when using Oracle GoldenGate.

Now that you have an understanding of the difference between ADD SCHEMATRANDATA and ADD TRANDATA, you will need to add it to database configuration through Oracle GoldenGate. To do this, you need to log in to the Oracle Database as the GoldenGate user through GGSCI. The steps for this process are outlined here.

1. Access the GGSCI.

[oracle@ggtest12c1 dirprm]$ $OGG\_HOME/ggsci  
  
Oracle GoldenGate Command Interpreter for Oracle  
Version 12.1.2.1.0 OGGCORE\_12.1.2.1.0\_PLATFORMS\_140727.2135.1\_FBO  
Linux, x64, 64bit (optimized), Oracle 12c on Aug  7 2014 10:21:34  
Operating system character set identified as UTF-8.  
  
Copyright (C) 1995, 2014, Oracle and/or its affiliates. All rights reserved.  
  
GGSCI (ggtest12c1.acme.com) 1>

1. Log in as the GoldenGate database user.

GGSCI (ggtest12c1.acme.com) 1> dblogin userid ggate password ggate  
Successfully logged into database.  
  
GGSCI (ggtest12c1.acme.com as ggate@src12c) 2>

1. Run the desired TRANDATA option. In our case we will use SCHEMATRANDATA because this an Oracle-to-Oracle setup.

GGSCI (ggtest12c1.acme.com as ggate@src12c) 4> add schematrandata scott  
  
2015-07-25 14:06:26 INFO OGG-01788 SCHEMATRANDATA has been added on schema scott.  
  
2015-07-25 14:06:26 INFO OGG-01976 SCHEMATRANDATA for scheduling columns has been added on schema scott.  
  
GGSCI (ggtest12c1.acme.com as ggate@src12c) 5>

Now you have everything in place that needs to be done to successfully configure the database. The next tasks you need to tackle are building the capture and data pump processes. The following sections describe how to edit parameter files and build the processes needed to make Oracle GoldenGate work.

### Edit Parameter Files

Before you can edit any parameter files, you need to be in the Oracle GoldenGate Home directory. Once you are in the $OGG\_HOME directory, start the GGSCI and issue the EDIT PARAMS command. This command is used to edit the parameter files associated with the processes running in the GoldenGate environment.

The steps to edit a parameter file from within Oracle GoldenGate are as follows:

1. Start GGSCI.

[oracle@ggtest12c1 ∼]$ cd $OGG\_HOME  
[oracle@ggtest12c1 oggcore\_1]$ ./ggsci  
  
Oracle GoldenGate Command Interpreter for Oracle  
Version 12.1.2.1.0 OGGCORE\_12.1.2.1.0\_PLATFORMS\_140727.2135.1\_FBO  
Linux, x64, 64bit (optimized), Oracle 12c on Aug  7 2014 10:21:34  
Operating system character set identified as UTF-8.  
  
Copyright (C) 1995, 2014, Oracle and/or its affiliates. All rights reserved.  
  
GGSCI (ggtest12c1.acme.com) 1>

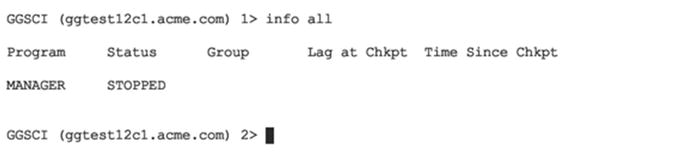
1. Edit the desired parameter file. Parameter files are named the same as the group or process name.

GGSCI (ggtest12c1.acme.com) 1> edit params esrc1

1. Enter the desired parameters, starting a new line with each parameter statement.

EXTRACT ESRC1  
USERID ggate, PASSWORD <pwd>  
TRANLOGOPTIONS DBLOGREADER  
SETENV (ORACLE\_HOME="/opt/app/oracle/product/12.1.0.2/dbhome\_1")  
SETENV (ORACLE\_SID="src12c")  
WARNLONGTRANS 1h, CHECKINTERVAL 30m  
EXTTRAIL ./dirdat/lt  
TABLE SCOTT.EMP;

The steps just listed are an example of editing a parameter file for an extract (capture) process. The same steps can be followed for any of the Oracle GoldenGate processes. At this point, if you were to execute an INFO ALL command from GGSCI, notice that the only process listed is the manager (MGR) process (Figure [3-3](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig3)). This is due to GoldenGate’s ability to allow simple file edits without the processes being in place. Allowing you to edit from within GGSCI ensures that the parameter files are located in the dirprm subdirectory under Oracle GoldenGate.



###### Figure 3-3. GGSCI with only MGR process adding the extract (capture) process

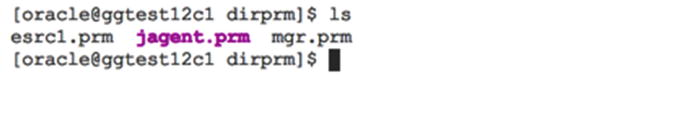
For anything in Oracle GoldenGate to work, you need to have a way of capturing the data that changes within the database. This is achieved by using an extract process; you might also hear the extract process called the capture process. There are two types of capture processes, commonly referred to as classic and integrated capture processes. For the simplicity of a unidirectional setup, we look at how to add a classic extract (capture) process.

In the previous section, you saw how to edit a parameter file from within the GGSCI. If you saved the file after editing, you should see a parameter file named esrc1.prm in the $OGG\_HOME/dirprm directory.

###### Note

Although I use numbers at the end of my process group names, it is not best practice to do this because the report files that are generated and associated with the process could be overwritten due to the number at the end.

Notice in the dirprm directory there are two other parameter files, jagent.prm and mgr.prm (see Figure [3-4](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig4)). Ignore these parameter files for now, as they are used by other processes within Oracle GoldenGate.



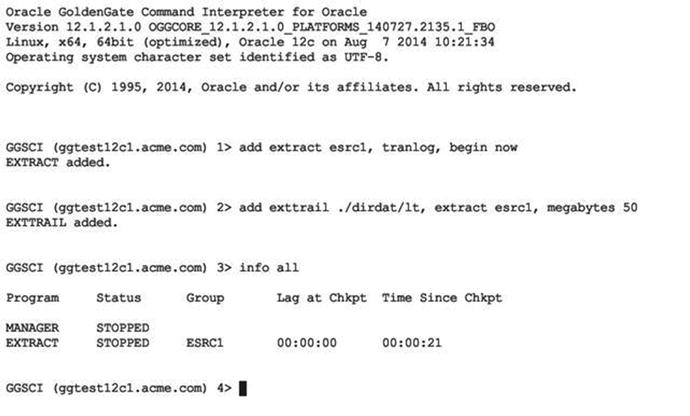
###### Figure 3-4. dirprm with parameter files

To add the extract process to Oracle GoldenGate, you need to run two commands (Listing [3-19](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1)) within GGSCI.

###### Listing 3-19. Adding an Extract Process

GGSCI> add extract esrc1, tranlog, begin now  
GGSCI> add exttrail ./dirdat/lt, extract esrc1, megabytes <size>

These two add commands will add a classic extract named esrc1 with a local trail file in the $OGG\_HOME/dirdat directory using the prefix lt (Figure [3-5](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig5)).



###### Figure 3-5. GGSCI after adding extract

Although the classic extract has been added, before you can start the process the underlying tables that will be replicated require additional configuration. This additional configuration is called TRANDATA. Adding TRANDATA ensures that all the tables that will be replicated have the required supplemental logging enabled. The settings that are added when running TRANDATA can be viewed when you look at the tables directly. They will have a supplemental log group added with a log name that is prefixed with a GG. This is an indicator that GoldenGate has turned on TRANDATA for the table. Once TRANDATA has been added to the tables that GoldenGate will capture from, you can start the process and verify that the local trail file is produced in the $OGG\_HOME/dirdat directory.

###### Note

I typically wait to start the extract (capture) process until I have built the data pump process. I do this to make sure that my network on the target side works and ensure that trail files can be shipped over the network.

### Starting the Extract

With everything in place, you should now be able to start the extract and begin capturing transactions into the local trail files located in the dirdat subdirectory.

To start the extract, issue the following command:

GGSCI (ggtest12c1.acme.com as ggate@src12c) 9> start extract esrc1

Once the extract has started, it can be confirmed using the INFO ALL command .

GGSCI (ggtest12c1.acme.com as ggate@src12c) 10> info all  
Program     Status      Group       Lag at Chkpt  Time Since Chkpt  
MANAGER     RUNNING  
EXTRACT     RUNNING     ESRC1       00:00:00      00:00:01

After confirming that the extract has started, you can exit GGSCI and verify the trail file was created in the $OGG\_HOME/dirdat directory, as illustrated in Listing [3-20](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1).

###### Listing 3-20. Confirming Local Trail Files

GGSCI (ggtest12c1.acme.com as ggate@src12c) 11> exit  
[oracle@ggtest12c1 dirprm]$ cd ../dirdat  
[oracle@ggtest12c1 dirdat]$ ls -ltr  
total 4  
-rw-r-----. 1 oracle oinstall 1437 Jul 25 14:53 lt000000

With the extract started, you can now focus on creating the data pump process.

### Extract (Data Pump) Process

With the capture process running, you are capturing any transactions that are occurring in the database at the time they are committed. This means that transactions are going into the local trail file in the order in which they are committed and ready to be shipped to the remote server in the architecture. To facilitate the shipping of the trail files, you need to create another extract that is commonly known as the data pump process.

The data pump process is a standard extract that is configured as a pass-through for shipping of the trail files. What makes this process different from a standard capture extract is that the parameter file is using a parameter called PASSTHRUto ensure that the trail file is shipped through the process to the remote location without any additional overhead. Listing [3-21](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1) shows you what a data pump file looks like.

###### Listing 3-21. Data Pump Parameter File

EXTRACT PSRC1  
PASSTHRU  
RMTHOST 10.10.1.12, MGRPORT 15000, COMPRESS  
RMTTRAIL ./dirdat/rt  
TABLE SCOTT.EMP;

Others things to be aware of in the data pump parameter file are the address for the remote host (RMTHOST) and the manager port (MGRPORT) number. The RMTHOSTparameter tells GoldenGate what server to access for shipping of trail files. The MGRPORTparameter tells the extract what manager process or port to connect to on the remote host. These parameters ensure that network traffic occurs between source and target systems.

###### Note

Although the PASSTHRU parameter is used to ship trail files without affecting them, there are times when you will want to interact with the trail file while shipping. In that case, the PASSTHRU parameter can be turned on and off using PASSTHRU and NOPASSTHRU.

#### Adding the Extract (Data Pump) Process

Adding the extract (data pump) process is done using the same methodology that is used when adding a capture (extract) process. You first have to edit the parameter file and make sure it is in the $OGG\_HOME/dirprm directory, then add the process using GGSCI commands. The following steps will help you add a data pump process to your configuration.

1. Edit the parameter file.

GGSCI (ggtest12c1.acme.com) 5> edit params psrc1

1. Add the extract (data pump) to the architecture through GGSCI. Notice that the extract is looking at the local trail files that the capture extract uses.

GGSCI (ggtest12c1.acme.com) 6> ADD EXTRACT PSRC1, EXTTRAILSOURCE ./dirdat/lt

1. Tell the extract (data pump) where to write the remote trail file. This is where the extract is sending the trail file along with renaming it on the remote side. Notice that the size of the remote trail file matches that of the local trail file.

GGSCI (ggtest12c1.acme.com) 7> ADD RMTTRAIL ./dirdat/rt, EXTRACT PSRC1, megabytes 50

#### Starting the Data Pump (Extract) Process

Once the process is added to the GGSCI environment, it can be reviewed and verified by issuing an INFO ALL command just as you did with the extract process previously. This lists all the processes associated with GoldenGate on the local server. Listing [3-22](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1) provides sample output .

###### Listing 3-22. Output After Adding Data Pump

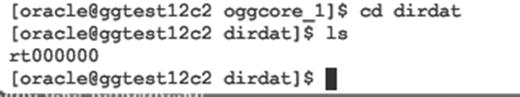
GGSCI (ggtest12c1.acme.com) 8> info all  
Program     Status      Group       Lag at Chkpt  Time Since Chkpt  
MANAGER     RUNNING  
EXTRACT     RUNNING     ESRC1       00:00:00      00:00:01  
EXTRACT     STOPPED     PSRC1       00:00:00      00:02:32

You will notice that the data pump is not started yet. To start the data pump, you need to issue the START EXTRACT command because data pumps are still extracts, just configured as a pass-through to allow for trail file shipping. Listing [3-23](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1) shows how to start a data pump process.

###### Listing 3-23. Starting Data Pump

GGSCI (ggtest12c1.acme.com) 9> start extract psrc1  
Sending START request to MANAGER ...  
EXTRACT PSRC1 starting

Once the process has started, it can be verified again using the INFO ALL command within GGSCI. Once the data pump is up successfully, you should be able to check the $OGG\_HOME/dirdat directory on the remote host as shown in Figure [3-6](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig6).



###### Figure 3-6. Verifying that the remote trail file is in $OGG\_HOME/dirdat (remote)

Now with the source side of the architecture running, capturing, and shipping transactions, you are almost finished setting up a unidirectional replication. Before you can complete the replicat configuration, though, you need to instantiate the environment. Let’s take a look at that now.

## Instantiation

The instantiation process is the process used to load a copy of static source data into the target database. This process captures a point-in-time snapshot of the data, and Oracle GoldenGate maintains the consistency by applying captured transactional data while the static data are loaded. Once instantiation is complete, Oracle GoldenGate will maintain the synchronization state throughout ongoing transactional changes.

Oracle GoldenGate provides three initial load methods specifically for Oracle databases:

1. Oracle Data Pump.
2. Direct Build Load to SQL\*Loader.
3. Load from an Input File to SQL\*Loader.

The most common approach for instantiation with an Oracle database is to use Oracle Data Pump, so we review this approach here in place of the other two approaches. Using this method, you start the extract, data pump, and replicat at the SCN at which the copy was made. This ensures that any transactions that were copied prior to that SCN are skipped to avoid collisions from integrity violations.

To instantiate using Oracle Data Pump, you need to have the extract and data pump running. Once the extract has been running for a few minutes, you can begin the export process with Oracle Data Pump. The key thing to remember is that you have to use the export data pump parameter FLASHBACK\_SCN.

###### Note

FLASHBACK\_SCN tells the Oracle database to make a consistent copy of all the data from that point in time.

The FLASHBACK\_SCN parameter is very important to the instantiation process. This is the same number that you will use to start the replicat once the import of the static data is complete. An example of a parameter file for Oracle Data Pump Export is provided in Listing [3-24](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1).

###### Listing 3-24. Export Data Pump Paramter Files

DUMPFILE="expdp\_%U.dmp"  
LOGFILE="expdp.log"  
DIRECTORY=EXPORTS  
PARALLEL=4  
JOB\_NAME='EXPDP\_GG\_FULL\_SCOTT'  
SCHEMA=SCOTT  
COMPRESSION=ALL  
CONTENT=ALL  
FLASHBACK\_SCN=1891898

###### Note

To grab the SCN to be used after starting the extract, issue the following SQL:

Select current\_scn from v$database;

With FLASHBACK\_SCN set in the export data pump parameter file, the export can be run and later imported into the target database. Once the data are imported on the target side, you can begin working on the replicat process.

## Apply (Replicat) Process

With the import complete, you can now create an apply (replicat) process that will process all the transactions that are stored in the trail files on the remote system. Adding a replicat process is just like adding your extracts earlier. The replicat is responsible for reading the remote trail files and applying the data found in chronological order. This ensures that the data are applied in the same order in which they were captured.

Just like the capture process, the replicat process has a few different modes in which it can be configured:

1. Classic .
2. Coordinated .
3. Integrated .

Any one of these modes can be used with any type of extract process because they represent different types of processes with the data. Because you have been reviewing the classic mode processes in this chapter, let’s stay with a classic mode replicat.

Just like the other Oracle GoldenGate processes, the replicat process uses a parameter file that needs to be configured. The parameter file can be created through GGSCI using the EDIT PARAMS command . Listing [3-25](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1) is an example of what a replicat parameter file could look like.

###### Listing 3-25. Replicat Parameter File

REPLICAT RTGT1  
SETENV (ORACLE\_HOME="/opt/app/oracle/product/12.1.0.2/dbhome\_1")  
SETENV (ORACLE\_SID="tgt12c")  
USERID ggate, PASSWORD ggate  
ASSUMETARGETDEFS  
DISCARDFILE ./dirrpt/RTGT1.dsc, append, megabytes 500  
map SCOTT.EMP, target SCOTT.EMP;

###### Note

There are two things to notice in this parameter file. The first is the clear text password for the GoldenGate user. This is only shown as an example; you should be using the new USERALIAS option in GoldenGate. The second is the DISCARDFILE. Starting in GoldenGate 12c (12.2), the discard file is automatically created for you if you do not specify one.

You will notice in the example that there are some similarities to the extract (capture) process and some differences. In this parameter file, the parameters ASSUMETARGETDEFS, DISCARDFILE, and MAP are telling the replicat process what it needs to do. The ASSUMETARGETDEFS parameters lets the replicat process know that the table structure on both sides (source and target) should be the same. DISCARDFILEis where the replicat process should place any transactions that have errors out of the apply process. Finally, the MAP parameter is doing the mapping between the transactions in the trail files and the target tables on the target system.

These basic parameters allow you to replicate data between the SCOTT.EMPtable on the source system and the SCOTT.EMP table on the target system. The MAP parameter can do a few more things, but they are not covered here because this is just a basic configuration.

### Adding the Apply (Replicat) Process

Just like the other processes, adding the apply (replicat) process is done from the GGSCI. The primary difference comes when the replicat is started. Let’s walk through adding the apply (replicat) process now.

1. Edit the parameter file.

GGSCI (ggtest12c2.acme.com) 2> edit params rtgt1

1. Add a checkpoint table for the replicat to keep track of applied transactions. Notice that you have to log in to the database from GGSCI and then run ADD CHECKPOINTTABLE. If the checkpoint table name is configured in the ./GLOBALS file, then GoldenGate uses that name instead of the default table name.

GGSCI (ggtest12c2.acme.com) 1> dblogin userid ggate password ggate  
Successfully logged into database.  
  
GGSCI (ggtest12c2.acme.com as ggate@rmt12c) 2> add checkpointtable  
  
No checkpoint table specified. Using GLOBALS specification (ggate.checkpoint)…  
  
Successfully created checkpoint table ggate.checkpoint.

1. Add the apply (replicat) to the architecture through GGSCI. Notice that the replicat is looking at the location of the remote trail files that were created earlier by the data pump process.

GGSCI (ggtest12c2.acme.com as ggate@rmt12c) 3> add replicat RTGT1, exttrail ./dirdat/rt  
REPLICAT added.

At this point, when you do an INFO ALL, you should see the manager process and the replicat process in the GGSCI as illustrated in Listing [3-26](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1). Now, you are ready to start the replicat.

###### Listing 3-26. GGSCI Output After Adding Replicat

GGSCI (ggtest12c2.acme.com as ggate@rmt12c) 4> info all  
Program     Status      Group       Lag at Chkpt  Time Since Chkpt  
MANAGER     RUNNING  
REPLICAT    STOPPED     RTGT1       00:00:00      00:03:07

### Starting the Apply (Replicat) Process

Although at this point you can start the apply (replicat) process and start replicating transactions, you need to pause for a minute. Earlier we discussed the SCN. Besides using the SCN for extracting the static data from the source, you need to use it here when starting the apply (replicat) process. This ensures that the replicat process is started at a point in time when transactions are not conflicting with each other and failing transactional integrity.

###### Note

If using Oracle Database 11.2.0.4 or later, ensure that enable\_goldengate\_replication is set to true.

To start the apply (replicat) process, you need to issue a command similar to the one shown in in Listing [3-27](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_3_Chapter.html#Fig1).

###### Listing 3-27. Starting Replicat

Start replicat rtgt1, [ aftercsn || atscn ] [ SCN ]

When running this command with the SCN you gathered earlier, the process attempts to start the replicat from that point in time. Let’s start the replicat now.

1. Issue start replicat with current\_scn.

GGSCI (ggtest12c2.acme.com as ggate@rmt12c) 5> start replicat rtgt1, aftercsn 1891898

1. Once the command is run, you can check the status of the replicat by issuing INFO ALL and monitoring the start of the replicat.

GGSCI (ggtest12c2.acme.com) 10> info all  
  
Program     Status      Group       Lag at Chkpt  Time Since Chkpt  
  
MANAGER     RUNNING  
REPLICAT    RUNNING     RTGT1       00:00:00      00:00:00

At this point, you should now be replicating transactions in a single direction between the source and target databases. This can be verified by doing any type of DML against the source database, in the schema being replicated, and seeing the changes reflected on the target side after a commit has occurred.

# 7. Advanced Features

To this point, you have been given a lot of information related to how Oracle GoldenGate can be installed, configured, run, verified and monitored. These are all great things to understand when running Oracle GoldenGate. Although these are the basics, Oracle GoldenGate also provides additional features to help you customize and make management a bit easier. These are the advanced features of Oracle GoldenGate.

Advanced features are used to help administrators customize, scale, and process transactions in a more robust way. This chapter takes a look at how to create and use macros, define and use tokens, and other advanced features of Oracle GoldenGate.

## Macros

Macros are a powerful feature of Oracle GoldenGate. In the many deployments I have done, this seems to be the one feature that can prevent many mistakes within the environment. Macros provide a way to modularize the code base for the replication environment and make it easier to move replication settings between release environments. Many people, however, do not understand this simple time-saving technique.

Although macros are a timesaver, the question is where these modules of code should be stored for reuse. If you look at the subdirectories within the Oracle GoldenGate home, you will notice there is no directory named dirmac. This is normal because most people who use macros often put the macro files in the dirprm directory. I actually think this is bad practice because it doesn’t provide a clear mapping of what is in the directory. To remedy this problem, you should create the dirmac” directory manually; then all the macros that you use within the environment can be called from a single location.

### Creating a Macro

As discussed, macros are a very powerful tool to use within Oracle GoldenGate, as they provide the ability to modularize the GoldenGate environment. To use a macro, it first needs to be created and then set up to be used within the GoldenGate processes.

A macro is simply a code block that can be reused over and over again in the same environment or copied to other environments. The basics of a macro read similar to a PL/SQL code block. Listing [7-1](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1) shows the basic components.

###### Listing 7-1. Basic Macro Code

MACRO <name>  
BEGIN  
<GoldenGate related information>  
END;

In Listing [7-1](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1), you see that the macro is created by simply editing a text file and adding lines that name the macro and define the beginning and the end of the macro. Everything in between the BEGIN and END calls of the macro are general GoldenGate commands that will be read when the GoldenGate process starts. To illustrate this functionality, Listing [7-2](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1) shows what the macro rtables look like in my test environment.

###### Listing 7-2. Test Macro Example

MACRO #rtables  
BEGIN  
MAP SOE.ADDRESSES, TARGET SOE.ADDRESSES;  
MAP SOE.CARD\_DETAILS, TARGET SOE.CARD\_DETAILS THREAD(2);  
MAP SOE.CUSTOMERS, TARGET SOE.CUSTOMERS, THREAD(3);  
MAP SOE.INVENTORIES, TARGET SOE.INVENTORIES, THREAD(4);  
MAP SOE.LOGON, TARGET SOE.LOGON, THREAD(5);  
MAP SOE.ORDER\_ITEMS, TARGET SOE.ORDER\_ITEMS, THREAD(1);  
MAP SOE.ORDERENTRY\_METADATA, TARGET SOE.ORDERENTRY\_METADATA, THREAD(2);  
MAP SOE.PRODUCT\_DESCRIPTIONS, TARGET SOE.PRODUCT\_DESCRIPTIONS, THREAD(4);  
MAP SOE.PRODUCT\_INFORMATION, TARGET SOE.PRODUCT\_INFORMATION, THREAD(5);  
MAP SOE.WAREHOUSES, TARGET SOE.WAREHOUSES, THREAD(1);  
END;

###### Note

The lines with THREAD(#) are related to a coordinated replicat.

Now that you have a macro that can be used to map tables, how do you use the macro in a parameter file? Like many programming languages that allow you to modularize code, an INCLUDE statement is needed to tell the GoldenGate process to read the associated file. Listing [7-3](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1) shows how this is done using the replicat process.

###### Listing 7-3. Illustration of Including Macros

**INCLUDE ./dirmac/rtables.mac**   
REPLICAT RSRC1  
SETENV (ORACLE\_HOME="/opt/app/oracle/product/12.1.0.2/dbhome\_1")  
SETENV (ORACLE\_SID="rmt12c")  
USERID GGATE, PASSWORD \*\*\*\*\*\*\*\*  
ASSUMETARGETDEFS  
REPORTCOUNT EVERY 5 SECONDS, RATE  
DISCARDFILE ./dirrpt/RSRC1.dsc, append, megabytes 500  
#rtables();

### Executing a Macro

In Listing [7-3](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1), you see that the INCLUDE statement is listed at the top of the parameter file. At the bottom of the parameter file, the macro that is desired is called. By providing these two pieces of information, you are telling GoldenGate that you want to read the macro and use it as part of the fundamental operation of the GoldenGate process.

When the GoldenGate process is started, the INCLUDE file is read (Listing [7-4](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1)) and the macro is executed to provide the process.

###### Listing 7-4. Execution During Startup of Process

INCLUDE ./dirmac/logon.mac  
MACRO #logon\_settings  
BEGIN  
USERID ggate, PASSWORD \*\*\*\*\*  
END;  
**INCLUDE ./dirmac/rtables.mac**   
**MACRO #rtables**   
**BEGIN**   
**MAP SOE.ADDRESSES, TARGET SOE.ADDRESSES;**   
**MAP SOE.CARD\_DETAILS, TARGET SOE.CARD\_DETAILS THREAD(2);**   
**MAP SOE.CUSTOMERS, TARGET SOE.CUSTOMERS, THREAD(3);**   
**MAP SOE.INVENTORIES, TARGET SOE.INVENTORIES, THREAD(4);**   
**MAP SOE.LOGON, TARGET SOE.LOGON, THREAD(5);**   
**MAP SOE.ORDER\_ITEMS, TARGET SOE.ORDER\_ITEMS, THREAD(1);**   
**MAP SOE.ORDERENTRY\_METADATA, TARGET SOE.ORDERENTRY\_METADATA, THREAD(2);**   
**MAP SOE.ORDERS, #runrates(3);**   
**MAP SOE.PRODUCT\_DESCRIPTIONS, TARGET SOE.PRODUCT\_DESCRIPTIONS, THREAD(4);**   
**MAP SOE.PRODUCT\_INFORMATION, TARGET SOE.PRODUCT\_INFORMATION, THREAD(5);**   
**MAP SOE.WAREHOUSES, TARGET SOE.WAREHOUSES, THREAD(1);**   
**END;**   
REPLICAT RSRC1  
SETENV (ORACLE\_HOME="/opt/app/oracle/product/12.1.0.2/dbhome\_1")  
SETENV (ORACLE\_SID="rmt12c")  
#logon\_settings()  
USERID ggate, PASSWORD \*\*\*\*\*  
ASSUMETARGETDEFS  
REPORTCOUNT EVERY 5 SECONDS, RATE  
DBOPTIONS NOSUPPRESSTRIGGERS  
REPERROR (DEFAULT, EXCEPTION)  
REPERROR (-1, EXCEPTION)  
REPERROR (-1403, EXCEPTION)  
REPERROR (-2291, EXCEPTION)  
DISCARDFILE ./dirrpt/RSRC1.dsc, append, megabytes 500  
**#rtables();**   
**MAP SOE.ADDRESSES, TARGET SOE.ADDRESSES;**   
**MAP SOE.CARD\_DETAILS, TARGET SOE.CARD\_DETAILS THREAD(2);**   
**MAP SOE.CUSTOMERS, TARGET SOE.CUSTOMERS, THREAD(3);**   
**MAP SOE.INVENTORIES, TARGET SOE.INVENTORIES, THREAD(4);**   
**MAP SOE.LOGON, TARGET SOE.LOGON, THREAD(5);**   
**MAP SOE.ORDER\_ITEMS, TARGET SOE.ORDER\_ITEMS, THREAD(1);**   
**MAP SOE.ORDERENTRY\_METADATA, TARGET SOE.ORDERENTRY\_METADATA, THREAD(2);**   
**MAP SOE.ORDERS, #runrates(3);**   
**MAP SOE.PRODUCT\_DESCRIPTIONS, TARGET SOE.PRODUCT\_DESCRIPTIONS, THREAD(4);**   
**MAP SOE.PRODUCT\_INFORMATION, TARGET SOE.PRODUCT\_INFORMATION, THREAD(5);**   
**MAP SOE.WAREHOUSES, TARGET SOE.WAREHOUSES, THREAD(1);**

As you can see from the examples, it is easy to create a macro, include it in a parameter file, and have it execute. The examples showed here are very basic for demonstration purposes; however, the real power of a macro comes into play when you can nest macros withing macros to make a very modular configuration.

## Tokens

Tokens are another advanced feature that can be used to capture and store data within the trail file’s user token area. Data that are stored as tokens can be used to customize the way that Oracle GoldenGate delivers information. A few ways that tokens can be used are:

* Column mappings.
* Used in stored procedures that are called by SQLEXEC.
* User exits.
* Macros.

###### Note

There are two types of tokens: user-defined and GoldenGate-specific tokens (GGSToken). GGSTokens are used to store LOGSCN and TRANID, among other internal information.

Because tokens can be used in a wide range of GoldenGate areas, these bits of user-defined information can be powerful for defining specifics of a business rule or what is happening with the GoldenGate environment.

### Defining Tokens

To define a token, you must define the token name and associate it with data that should be captured; this is normally environment-related information. The data that can be defined within a token can be any valid character data or values retrieved from an Oracle GoldenGate column-conversion function.

The record header of the trail file permits up to 2,000 bytes of data to be stored for user token information. The token name, length of data, and the data itself must all fit into the 2,000 bytes allocated in the record header.

You can define a token in the capture and pump processes of the GoldenGate environment. The TOKENS option is part of the TABLE parameter for the extract process. Listing [7-5](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1) highlights the syntax needed for a token definition.

###### Listing 7-5. Token Definition Syntax

TABLE <table defined>, TOKENS (<token name> = <token\_data> [, ...]);

The syntax for defining a token can be placed either directly into the parameter file for the extract or it can be defined in a macro for reuse.

### Running Tokens

Now that you know how to define a token, putting them to use is the next step. As previously mentioned, tokens are defined as part of the TABLE parameter in the extract process. This allows you to define customer user tokens. To illustrate how to define a token in an extract, Listing [7-6](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1) shows how this is done per table.

###### Listing 7-6. Defining Tokens in Extract Process

INCLUDE ./dirmac/logon.mac  
--CHECKPARAMS  
EXTRACT ESRC1  
#logon\_settings()  
TRANLOGOPTIONS DBLOGREADER  
SETENV (ORACLE\_HOME="/opt/app/oracle/product/12.1.0.2/dbhome\_1")  
SETENV (ORACLE\_SID="src12c")  
EXTTRAIL ./dirdat/lt  
WARNLONGTRANS 1h, CHECKINTERVAL 30m  
WILDCARDRESOLVE IMMEDIATE  
REPORTCOUNT EVERY 3 MINUTES, RATE  
INCLUDE ./dirmac/heartbeat\_extract.mac  
**TABLE SOE.ADDRESSES,TOKENS (**   
**TK-OSUSER = @GETENV ('GGENVIRONMENT' , 'OSUSERNAME'),**   
**TK-GROUP = @GETENV ('GGENVIRONMENT' , 'GROUPNAME'),**   
**TK-HOST =  @GETENV('GGENVIRONMENT' , 'HOSTNAME'),**   
TABLE SOE.CARD\_DETAILS;  
TABLE SOE.CUSTOMERS;  
TABLE SOE.INVENTORIES;  
TABLE SOE.LOGON;  
TABLE SOE.ORDER\_ITEMS;  
TABLE SOE.ORDERENTRY\_METADATA;  
TABLE SOE.ORDERS;  
TABLE SOE.PRODUCT\_DESCRIPTIONS;  
TABLE SOE.PRODUCT\_INFORMATION;  
TABLE SOE.WAREHOUSES;

In Listing [7-6](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1), notice that there are four tokens defined. These tokens start with the prefix TK-.

###### Note

A token can be named anything you want to name it.

These are the names that will be used on the replicat side when reading the token from the trail file being processed. Also notice that the information being gathered is coming from the environment that GoldenGate is running in by using the @GETENV parameter. The information that is obtained from @GETENV is stored as the value of the token with the prefix of TK-. At a later point in the replication, this information can be retrieved by calling the token name.

###### Note

Tokens can be populated with any type of information; if you need more information on the @GETENV function, you can find it at [http://​docs.​oracle.​com/​goldengate/​c1221/​gg-winux/​GWURF/​column\_​conversion\_​functions015.​htm#GWURF788](http://docs.oracle.com/goldengate/c1221/gg-winux/GWURF/column_conversion_functions015.htm#GWURF788).

### Applying Tokens

With tokens being defined in the extract process and having your desired information captured in user tokens, you will want to have this information applied on the receiving end of replication. To do this, the replicat has to be told about the tokens and how they should be applied in the database.

To apply tokens, they need to be mapped to columns in the target table. The target table can either be the replicat table with additional columns for tokens or a separate table with columns that only map the tokens. Either way, the tokens have to be mapped to columns. In Listing [7-7](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1), you will see how I map the table SOE.ADDRESS to a table that will only store the contents of the tokens.

###### Listing 7-7. Replicat Mapping with Tokens Via Macro

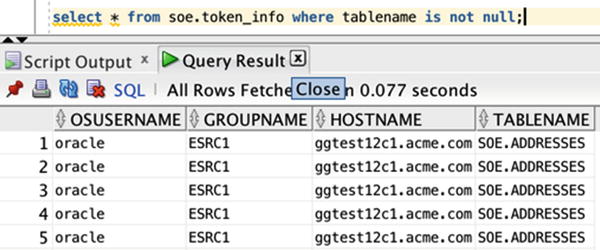
MACRO #rtables  
BEGIN  
**MAP SOE.ADDRESSES, TARGET SOE.ADDRESSES;**   
**MAP SOE.ADDRESSES, TARGET SOE.TOKEN\_INFO, COLMAP(OSUSERNAME=@TOKEN('TK-OSUSER'),GROUPNAME=@TOKEN('TK-GROUP'),HOSTNAME=@TOKEN('TK-HOST'),TABLENAME=@TOKEN('TK-TABLE'));**   
MAP SOE.CARD\_DETAILS, TARGET SOE.CARD\_DETAILS THREAD(2);  
MAP SOE.CUSTOMERS, TARGET SOE.CUSTOMERS, THREAD(3);  
MAP SOE.INVENTORIES, TARGET SOE.INVENTORIES, THREAD(4);  
MAP SOE.LOGON, TARGET SOE.LOGON, THREAD(5);  
MAP SOE.ORDER\_ITEMS, TARGET SOE.ORDER\_ITEMS, THREAD(1);  
MAP SOE.ORDERENTRY\_METADATA, TARGET SOE.ORDERENTRY\_METADATA, THREAD(2);  
MAP SOE.PRODUCT\_DESCRIPTIONS, TARGET SOE.PRODUCT\_DESCRIPTIONS, THREAD(4);  
MAP SOE.PRODUCT\_INFORMATION, TARGET SOE.PRODUCT\_INFORMATION, THREAD(5);  
MAP SOE.WAREHOUSES, TARGET SOE.WAREHOUSES, THREAD(1);  
END;

The key thing to notice in this MAP statement is that the TARGET table is SOE.TOKEN\_INFO. These tables are different in structure; however, the COLMAP option allows you to map what information is getting applied to what columns. In this case, I want to map all the tokens previously defined to the columns in the SOE.TOKEN\_INFO table. If you mapped this out logically in a table, it would look similar to Table [7-1](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1).

###### Table 7-1. Example of Column Mapping for Tokens

| **Column Name** | **Token Mapping** |
| --- | --- |
| OSUSERNAME | @TOKEN=(‘TK-OSUSER’) |
| GROUPNAME | @TOKEN=(‘TK-GROUP’) |
| HOSTNAME | @TOKEN=(‘TK-HOST’) |
| TABLENAME | @TOKEN=(‘TK-TABLE’) |

When the replicat begins to apply the information in the trail files to the database, the tokens will be mapped to the corresponding columns for SOE.TOKEN\_INFO. After the apply process is complete, you can query the SOE.TOKEN\_INFO table and verify that the token information has been applied to the table. Figure [7-1](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Fig1) shows the token information applied to the correct table.



###### Figure 7-1. Token info in the table

As you can see, the transactions that ran against the SOE.ADDRESSES table have presented the token information for the OS User who ran the GoldenGate process, the GoldenGate process group, the hostname where the token was defined, and the table that it was captured from.

Just like macros, tokens are a very powerful tool that can be used to customize the replication process. Tokens provide a way for you to define and capture custom or granular detail information from the source database and retain a record of it on the target side. Tokens also allow you to execute defined procedures using SQLEXEC depending on the contents of the token.

## Heartbeat

Another feature that is part of Oracle GoldenGate is the ability to track the latency within the replication framework by using a heartbeat process. A heartbeat process is set of GoldenGate processes that are used to calculate the “true” lag within the network, from end to end. From an administration standpoint, this is a handy tool for validating the lag in the network.

Starting in Oracle GoldenGate 12c (12.2.0.1.0), there are two types of heartbeat processes. The first type is the traditional heartbeat, which is external to the existing GoldenGate processes. The second type is the integrated heartbeat, which is new in 12.2.0.1.0. Let’s take a look at both processes so you can get an understanding of how they work.

### Traditional Heartbeat

The traditional heartbeat is a process that was developed to measure lag between the source and target by using “heartbeat” tables. Although there are tables required for this heartbeat process, there are a few different parts that need to be configured to make the process work.

To successfully set the traditional heartbeat structure within a GoldenGate environment, there are a number of objects that need to be set up. Table [7-2](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab2) provides a brief breakdown of what needs to be added to the source and target database to run a traditional heartbeat.

###### Table 7-2. Traditional Heartbeat Components

| **Location** | **Component** | **Type** |
| --- | --- | --- |
| Source | HEARTBEAT | Table |
| Source | SEQ\_GGS\_HEARTBEAT\_ID | Sequence |
| Source | HEARTBEAT\_TRIG | Trigger |
| Source | <no name required> | Scheduler Job |
| Target | GGS\_HEARTBEAT\_HISTORY | Table |
| Target | SEQ\_GGS\_HEARTBEAT\_HIST | Sequence |
| Target | GGS\_HEARTBEAT\_TRIG\_HIST | Trigger |
| Target | TOTAL\_LAG\_HB | View |

###### Note

The component names in Table [7-2](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab2) can be changed when creating your heartbeat framework. The names are mostly taken from documentation and personal setup information.

The original heartbeat information provided by Oracle can be found in My Oracle Support Note ID 1299679.1.

As you can tell, there are a few things that need to be configured to make the traditional heartbeat work. When you first start configuring traditional heartbeats, the components in Table [7-2](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab2) are just the basics. These components do not contain anything that Oracle GoldenGate will need to replicate information. Table [7-2](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab2) only provides the framework from a database level. Let’s take a closer look at this SQL framework before you dive into the specifics at the GoldenGate process level.

#### Source Database Configuration

The first thing that needs to be done is to set up the traditional heartbeat database components in the source database. These components consist of a table specific to heartbeat information, a sequence, a trigger, and a job scheduler configuration that fires the trigger. Let’s take a look at each of these components.

##### Traditional Heartbeat Table

The traditional heartbeat table is just a standard table that you can create in any schema that you like. Traditionally, the heartbeat table should be created in the schema that will house your Oracle GoldenGate objects. Listing [7-8](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1) shows the DDL for building out the table.

###### Note

Your heartbeat table can be different from this framework.

###### Listing 7-8. Create DDL for Traditional Heartbeat Table

drop table &&ogg\_user..heartbeat;  
  
-- Create table statement  
CREATE TABLE &&ogg\_user..HEARTBEAT  
(ID NUMBER ,  
SRC\_DB           VARCHAR2(30),  
EXTRACT\_NAME     varchar2(8),  
SOURCE\_COMMIT    TIMESTAMP,  
TARGET\_COMMIT    TIMESTAMP,  
CAPTIME          TIMESTAMP,  
CAPLAG           NUMBER,  
PMPTIME          TIMESTAMP,  
PMPGROUP         VARCHAR2(8 BYTE),  
PMPLAG           NUMBER,  
DELTIME          TIMESTAMP,  
DELGROUP         VARCHAR2(8 BYTE),  
DELLAG           NUMBER,  
TOTALLAG         NUMBER,  
thread           number,  
update\_timestamp timestamp,  
EDDLDELTASTATS   number,  
EDMLDELTASTATS   number,  
RDDLDELTASTATS   number,  
RDMLDELTASTATS   number,  
CONSTRAINT HEARTBEAT\_PK PRIMARY KEY (SRC\_DB)  
)  
/

Notice that the table will be used to keep track of key metric information and that the primary key is a standard number data type. This is due to the table being ideal for databases that are version 11g and below. If you build a heartbeat table in Oracle Database 12c, you can leverage the identity column features of the database as well. Now that you have an 11g version of the table created, you will need a sequence and a trigger to increment the information stored in the table. The parts of the traditional heartbeat can be created with the examples in Listing [7-9](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1).

###### Note

The PL/SQL code presented here is a commented set of code that can be optained from an Oracle White Paper at <http://www.ateam-oracle.com/wp-content/uploads/2013/04/OGG-Best-Practice-heartbeat-table-using-DBMS_SCHEDULER-V11_0-ID1299679.1.pdf>.

###### Listing 7-9. Sequence and Trigger Required

DROP SEQUENCE &&ogg\_user..SEQ\_GGS\_HEARTBEAT\_ID ;  
  
CREATE SEQUENCE &&ogg\_user..SEQ\_GGS\_HEARTBEAT\_ID INCREMENT BY 1 START WITH 1 ORDER ;  
  
CREATE OR REPLACE TRIGGER &&ogg\_user..HEARTBEAT\_TRIG  
BEFORE INSERT OR UPDATE ON &&ogg\_user..HEARTBEAT  
FOR EACH ROW  
BEGIN  
select SEQ\_GGS\_HEARTBEAT\_ID.nextval  
into :NEW.ID  
from dual;  
select systimestamp  
into :NEW.target\_COMMIT  
from dual;  
select trunc(to\_number(substr((:NEW.CAPTIME - :NEW.SOURCE\_COMMIT ),1, instr(:NEW.CAPTIME - :NEW.SOURCE\_COMMIT,' ')))) \* 86400  
+ to\_number(substr((:NEW.CAPTIME - :NEW.SOURCE\_COMMIT), instr((:NEW.CAPTIME - :NEW.SOURCE\_COMMIT),' ')+1,2)) \* 3600  
+ to\_number(substr((:NEW.CAPTIME - :NEW.SOURCE\_COMMIT), instr((:NEW.CAPTIME - :NEW.SOURCE\_COMMIT),' ')+4,2) ) \* 60  
+ to\_number(substr((:NEW.CAPTIME - :NEW.SOURCE\_COMMIT), instr((:NEW.CAPTIME - :NEW.SOURCE\_COMMIT),' ')+7,2))  
+ to\_number(substr((:NEW.CAPTIME - :NEW.SOURCE\_COMMIT), instr((:NEW.CAPTIME - :NEW.SOURCE\_COMMIT),' ')+10,6)) / 1000000  
into :NEW.CAPLAG  
from dual;  
select trunc(to\_number(substr((:NEW.PMPTIME - :NEW.CAPTIME),1, instr(:NEW.PMPTIME - :NEW.CAPTIME,' ')))) \* 86400  
+ to\_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME - :NEW.CAPTIME),' ')+1,2)) \* 3600  
+ to\_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME - :NEW.CAPTIME),' ')+4,2) ) \* 60  
+ to\_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME - :NEW.CAPTIME),' ')+7,2))  
+ to\_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME - :NEW.CAPTIME),' ')+10,6)) / 1000000  
into :NEW.PMPLAG  
from dual;  
select trunc(to\_number(substr((:NEW.DELTIME - :NEW.PMPTIME),1, instr(:NEW.DELTIME - :NEW.PMPTIME,' ')))) \* 86400  
+ to\_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME - :NEW.PMPTIME),' ')+1,2)) \* 3600  
+ to\_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME - :NEW.PMPTIME),' ')+4,2) ) \* 60  
+ to\_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME - :NEW.PMPTIME),' ')+7,2))  
+ to\_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME - :NEW.PMPTIME),' ')+10,6)) / 1000000  
into :NEW.DELLAG  
from dual;  
select trunc(to\_number(substr((:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT),1, instr(:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT,' ')))) \* 86400  
+ to\_number(substr((:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT), instr((:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT),' ')+1,2)) \* 3600  
+ to\_number(substr((:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT), instr((:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT),' ')+4,2) ) \* 60  
+ to\_number(substr((:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT), instr((:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT),' ')+7,2))  
+ to\_number(substr((:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT), instr((:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT),' ')+10,6)) / 1000000  
into :NEW.TOTALLAG  
from dual;  
end ;  
/

After creating the components in Listing [7-10](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1), the table needs to be primed to ensure that all the components are working. This is done by simply inserting into the table an initial record (Listing [7-10](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1)). Once this information is inserted, it should validate that the trigger fires and updates key information.

###### Listing 7-10. Insert Statement for Heartbeat

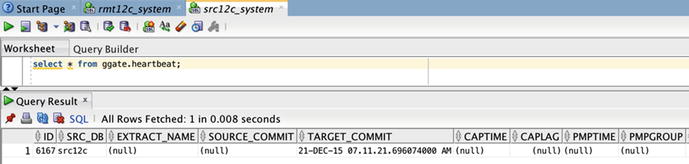
-- this assumes that the table is empty  
INSERT INTO &&ogg\_user..HEARTBEAT (SRC\_DB) select db name from V$database;  
commit;

Now that the source side of the heartbeat is configured, the next thing that needs to happen is to ensure this table is updated on a regular basis. To do this, you will need to set up a scheduler job that will fire the trigger every minute or on a defined timeframe that you are comfortable with. Listing [7-11](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1) provides you with a scheduler job normally used for heartbeat tables.

###### Listing 7-11. Schedule Job Setup

grant select on v\_$database to &&ogg\_user;  
  
BEGIN  
    SYS.DBMS\_SCHEDULER.DROP\_JOB(job\_name => '&&ogg\_user..OGG\_HB',  
                                defer => false,  
                                force => false);  
END;  
/  
  
CREATE OR REPLACE PROCEDURE &&ogg\_user..gg\_update\_hb\_tab IS  
v\_thread\_num     NUMBER;  
v\_db\_unique\_name VARCHAR2 (128);  
BEGIN  
SELECT db\_unique\_name  
INTO  v\_db\_unique\_name  
FROM v$database;  
  
UPDATE &&ogg\_user..heartbeat  
SET update\_timestamp = SYSTIMESTAMP  
,src\_db = v\_db\_unique\_name;  
END;  
/  
  
BEGIN  
SYS.DBMS\_SCHEDULER.CREATE\_JOB (  
job\_name => '&&ogg\_user..OGG\_HB',  
job\_type => 'STORED\_PROCEDURE',  
job\_action => '&&ogg\_user..GG\_UPDATE\_HB\_TAB',  
number\_of\_arguments => 0,  
start\_date => NULL,  
repeat\_interval => 'FREQ=MINUTELY',  
end\_date => NULL,  
job\_class => '"SYS"."DEFAULT\_JOB\_CLASS"',  
enabled => FALSE,  
auto\_drop => FALSE,  
comments => 'GoldenGate',  
credential\_name => NULL,  
destination\_name => NULL);  
  
SYS.DBMS\_SCHEDULER.SET\_ATTRIBUTE(  
name => '&&ogg\_user..OGG\_HB',  
attribute => 'restartable', value => TRUE);  
  
SYS.DBMS\_SCHEDULER.SET\_ATTRIBUTE(  
name => '&&ogg\_user..OGG\_HB',  
attribute => 'logging\_level', value => DBMS\_SCHEDULER.LOGGING\_OFF);  
  
SYS.DBMS\_SCHEDULER.enable(  
name => '&&ogg\_user..OGG\_HB');  
END;  
/

You will notice that this scheduler job will update the heartbeat table; however, the heartbeat table will have a few columns with NULL values (Figure [7-2](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Fig2)). This is normal because the trigger doesn’t have all the information needed. As the changes in the table are captured and replicated, the additional information will be provided on the target side.



###### Figure 7-2. Ouput of heartbeat table

#### Target Database Configuration

With the source side set up, you need to set up the target side. The target side configuration has to be set up the same way using SQL. The components needed for the heartbeat to work are similar to those for the source side: a table, a sequence, and a trigger.

The table that needs to be created is a simple table to keep track of all the historical information for the heartbeat. Historical information provides a way to validate the lag over time. Listing [7-12](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1) provides the DDL for the table.

###### Listing 7-12. Target Side Heartbeat Table

DROP TABLE &&ogg\_user..GGS\_HEARTBEAT\_HISTORY;  
  
CREATE TABLE &&ogg\_user..GGS\_HEARTBEAT\_HISTORY  
(      ID NUMBER ,  
SRC\_DB           VARCHAR2(30),  
EXTRACT\_NAME     varchar2(8),  
SOURCE\_COMMIT    TIMESTAMP,  
TARGET\_COMMIT    TIMESTAMP,  
CAPTIME          TIMESTAMP,  
CAPLAG           NUMBER,  
PMPTIME          TIMESTAMP,  
PMPGROUP         VARCHAR2(8 BYTE),  
PMPLAG           NUMBER,  
DELTIME          TIMESTAMP,  
DELGROUP         VARCHAR2(8 BYTE),  
DELLAG           NUMBER,  
TOTALLAG         NUMBER,  
thread           number,  
update\_timestamp timestamp,  
EDDLDELTASTATS   number,  
EDMLDELTASTATS   number,  
RDDLDELTASTATS   number,  
RDMLDELTASTATS   number  
);

You will notice that the table structure is the same as the source table. Like many other tables that are replicated in GoldenGate, having a matching table structure helps to simplify the replication requirements. Keeping this in mind, you will need to also create a sequence for the table to use (Listing [7-13](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1)).

###### Listing 7-13. Sequence Needed for Target Heartbeat Table

DROP SEQUENCE &&ogg\_user..SEQ\_GGS\_HEARTBEAT\_HIST ;  
CREATE SEQUENCE &&ogg\_user..SEQ\_GGS\_HEARTBEAT\_HIST INCREMENT BY 1 START WITH 1 ORDER ;

Finally, you need to create the trigger that will fire when transactions happen against the table. This trigger, just like in the source table, will do a few calculations against the incoming data and calculate the lag from source to target. Listing [7-14](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1) provides the DDL for the triggers.

###### Listing 7-14. Target Side Triggers

create or replace TRIGGER &&ogg\_user.GGS\_HEARTBEAT\_TRIG\_HIST  
BEFORE INSERT OR UPDATE ON ggate.GGS\_HEARTBEAT\_HISTORY  
FOR EACH ROW  
BEGIN  
select seq\_ggs\_HEARTBEAT\_HIST.nextval into :NEW.ID  
from dual;  
select systimestamp into :NEW.target\_COMMIT from dual;  
select trunc(to\_number(substr((:NEW.CAPTIME - :NEW.SOURCE\_COMMIT ),1, instr(:NEW.CAPTIME - :NEW.SOURCE\_COMMIT,' ')))) \* 86400  
+ to\_number(substr((:NEW.CAPTIME - :NEW.SOURCE\_COMMIT), instr((:NEW.CAPTIME - :NEW.SOURCE\_COMMIT),' ')+1,2)) \* 3600  
+ to\_number(substr((:NEW.CAPTIME - :NEW.SOURCE\_COMMIT), instr((:NEW.CAPTIME - :NEW.SOURCE\_COMMIT),' ')+4,2) ) \* 60  
+ to\_number(substr((:NEW.CAPTIME - :NEW.SOURCE\_COMMIT), instr((:NEW.CAPTIME - :NEW.SOURCE\_COMMIT),' ')+7,2))  
+ to\_number(substr((:NEW.CAPTIME - :NEW.SOURCE\_COMMIT), instr((:NEW.CAPTIME - :NEW.SOURCE\_COMMIT),' ')+10,6)) / 1000000  
into :NEW.CAPLAG  
from dual;  
select trunc(to\_number(substr((:NEW.PMPTIME - :NEW.CAPTIME),1, instr(:NEW.PMPTIME - :NEW.CAPTIME,' ')))) \* 86400  
+ to\_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME - :NEW.CAPTIME),' ')+1,2)) \* 3600  
+ to\_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME - :NEW.CAPTIME),' ')+4,2) ) \* 60  
+ to\_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME - :NEW.CAPTIME),' ')+7,2))  
+ to\_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME - :NEW.CAPTIME),' ')+10,6)) / 1000000  
into :NEW.PMPLAG  
from dual;  
select trunc(to\_number(substr((:NEW.DELTIME - :NEW.PMPTIME),1, instr(:NEW.DELTIME - :NEW.PMPTIME,' ')))) \* 86400  
+ to\_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME - :NEW.PMPTIME),' ')+1,2)) \* 3600  
+ to\_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME - :NEW.PMPTIME),' ')+4,2) ) \* 60  
+ to\_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME - :NEW.PMPTIME),' ')+7,2))  
+ to\_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME - :NEW.PMPTIME),' ')+10,6)) / 1000000  
into :NEW.DELLAG  
from dual;  
select trunc(to\_number(substr((:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT),1, instr(:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT,' ')))) \* 86400  
+ to\_number(substr((:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT), instr((:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT),' ')+1,2)) \* 3600  
+ to\_number(substr((:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT), instr((:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT),' ')+4,2) ) \* 60  
+ to\_number(substr((:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT), instr((:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT),' ')+7,2))  
+ to\_number(substr((:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT), instr((:NEW.TARGET\_COMMIT - :NEW.SOURCE\_COMMIT),' ')+10,6)) / 1000000  
into :NEW.TOTALLAG  
from dual;  
end ;  
/  
ALTER TRIGGER &&ogg\_user..GGS\_HEARTBEAT\_TRIG\_HIST ENABLE;

Now that the SQL components of the traditional heartbeat are in place, you will need to configure the GoldenGate processes for the heartbeat.

#### Heartbeat Configuration

To configure the heartbeat configuration is just like setting up normal replication; the only difference here is that the heartbeat table is the only object being replicated across the extracts and replicats. The design of this keeps the heartbeat as minimal as possible to prevent it from affecting existing replication and provide near real-time information on the network lag.

To do this, you will need to have three parameter files created, one parameter file for each of the processes needed in the heartbeat replication. Listing [7-15](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Par72) provides a breakdown of the parameter files and their contents.

Extract:

###### Listing 7-15. Heartbeat Parameter Files and Contents (Extract/Pump/Replicat)

INCLUDE ./dirmac/logon.mac  
--CHECKPARAMS  
EXTRACT EXT\_HB  
#logon\_settings()  
TRANLOGOPTIONS DBLOGREADER  
SETENV (ORACLE\_HOME="/opt/app/oracle/product/12.1.0.2/dbhome\_1")  
SETENV (ORACLE\_SID="src12c")  
WARNLONGTRANS 1h, CHECKINTERVAL 30m  
EXTTRAIL ./dirdat/h1  
--WILDCARDRESOLVE IMMEDIATE  
REPORTCOUNT EVERY 5 MINUTES, RATE  
INCLUDE ./dirmac/heartbeat\_extract.mac

Pump (Extract):

INCLUDE ./dirmac/logon.mac  
EXTRACT PMP\_HB  
#logon\_settings()  
RMTHOST 10.10.1.12, MGRPORT 15000, COMPRESS  
RMTTRAIL ./dirdat/h2  
INCLUDE ./dirmac/heartbeat\_pump.mac

Replicat:

INCLUDE ./dirmac/logon.mac  
REPLICAT REP\_HB  
SETENV (ORACLE\_HOME="/opt/app/oracle/product/12.1.0.2/dbhome\_1")  
SETENV (ORACLE\_SID="rmt12c")  
#logon\_settings()  
ASSUMETARGETDEFS  
REPORTCOUNT EVERY 5 MINUTES, RATE  
DISCARDFILE ./dirrpt/RSRC1.dsc, append, megabytes 500  
DBOPTIONS NOSUPPRESSTRIGGERS  
INCLUDE ./dirmac/heartbeat\_replicat.mac

You will notice that in each of these parameter files, there is a call to a macro for its specific replication information. The macro for each process is specific to providing the tokens needed for replication of the heartbeat information. Listing [7-16](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Par79) provides the information that is in these macros.

Extract:

###### Listing 7-16. Macro Information for Heartbeat

--Heartbeat Extract Macro  
--Heartbeat Table  
--Inital write - 12-21-2013 - BLC  
table ggate.heartbeat,  
tokens(  
capgroup=@getenv('GGENVIRONMENT','GROUPNAME'),  
captime=@DATE('YYYY-MM-DD HH:MI:SS.FFFFFF','JTS',@GETENV('JULIANTIMESTAMP')),  
eddldeltastats=@getenv ('DELTASTATS', 'DDL'),  
edmldeltastats=@getenv ('DELTASTATS', 'DML')  
);

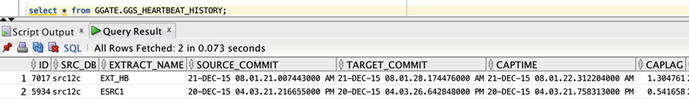
Pump (extract):

--Heartbeat Pump Macro  
--Heartbeat Table  
--Inital write - 12-21-2013 - BLC  
table ggate.heartbeat,  
tokens(  
pmpgroup=@getenv('GGENVIRONMENT','GROUPNAME'),  
pmptime=@DATE('YYYY-MM-DD HH:MI:SS.FFFFFF','JTS',@GETENV('JULIANTIMESTAMP'))  
);

Replicat:

--Heartbeat Replicat Macro  
--Tar\_Heartbeat Table  
--Inital write - 12-21-2013 - BLC  
MAP ggate.HEARTBEAT, TARGET ggate.GGS\_HEARTBEAT\_HISTORY,  
KEYCOLS (DELGROUP),  
INSERTMISSINGUPDATES,  
COLMAP (USEDEFAULTS,  
ID = 0,  
SOURCE\_COMMIT=@GETENV ('GGHEADER', 'COMMITTIMESTAMP'),  
EXTRACT\_NAME=@TOKEN ('CAPGROUP'),  
CAPTIME=@TOKEN ('CAPTIME'),  
PMPGROUP=@TOKEN ('PMPGROUP'),  
PMPTIME=@TOKEN ('PMPTIME'),  
DELGROUP=@GETENV ('GGENVIRONMENT', 'GROUPNAME'),  
DELTIME=@DATE ('YYYY-MM-DD HH:MI:SS.FFFFFF','JTS',@GETENV ('JULIANTIMESTAMP')),  
EDDLDELTASTATS=@TOKEN ('EDDLDELTASTATS'),  
EDMLDELTASTATS=@TOKEN ('EDMLDELTASTATS'),  
RDDLDELTASTATS=@GETENV ('DELTASTATS', 'DDL'),  
RDMLDELTASTATS=@GETENV ('DELTASTATS', 'DML')  
);

With everything in place now, you can start create and start the extract, pump, and replicat processes associated with the heartbeat process. As the schedule job begins to update the table on the source side, the information will be replicated and applied to the target side. All the information captured in the tokens will be inserted into the table on the target side, providing you with near real-time information on the network lag. Figure [7-3](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Fig3) provides a view into the table on the target side.



###### Figure 7-3. Output of the heartbeat process

Another nice feature of the heartbeat process is that if you use the macros as pointed out here, you can then integrate the heartbeat into your existing GoldenGate processes. This provides you a way to calculate the lag on processes replicating your data. This approach is shown in Figure [7-3](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Fig3) as well.

### Integrated Heartbeat

The heartbeat process has become such a main component of many GoldenGate environments, Oracle decided to take the process a step further and make it a core feature in the latest release of Oracle GoldenGate 12c. Starting with GoldenGate 12.2.0.1.0, the heartbeat process is now integrated into the core code base. This is both a good thing and a bad thing in my opinion. I say this because the new integrated heartbeat only solves the issue of the heartbeat in your existing processes; it does not allow you to monitor network lag like the traditional heartbeat can. In the end, the integrated heartbeat is a great feature if you want to quickly set up and monitor lag on your existing replication processes.

#### Parameters for Integrated Heartbeat

To set up the integrated heartbeat requires minimal configuration; however, like other aspects of GoldenGate, this setup requires adding a parameter within the configuration. If you want the parameters to take effect at the global level, the changes need to be made in the GLOBALS file. If the changes are to be localized to the process, they can be added to the individual process parameter files. The parameters that have to be used are the following:

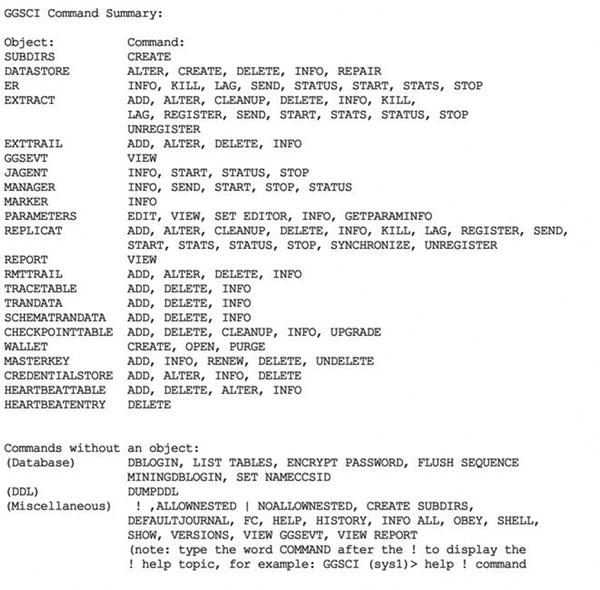
* HEARTBEATTABLE <table\_name>: This parameter allows you to define the heartbeat table you want to use; the default name will be GG\_HEARTBEAT.
* ENABLE\_HEARTBEAT\_TABLE | DISABLE\_HEARTBEAT\_TABLE: These parameters are used to either start or stop the heartbeat-related items in the GoldenGate environment. (The parameters can be used in either GLOBALS, Extract, or Replicat to enable or disable the heartbeat.)

The HEARTBEATTABLE parameter tells GoldenGate where to build and look for the heartbeat table. This parameter is normally placed in the GLOBALS file so when it is built using the ADD HEARTBEATTABLE command, GoldenGate automatically builds the table. After the table is built, it will be referenced based on the information in the GLOBALS file.

The next set of parameters that is used with the integrated heartbeat table are ENABLE\_HEARTBEAT\_TABLE and DISABLE\_HEARTBEAT\_TABLE. These parameters are designed to start and stop the heartbeat process in the processes. These parameters can either be added to the GLOBALS file or localized in the process parameter files.

#### Set Up Integrated Heartbeat

Once you have decided where the parameters for integrated heartbeat should be located, the next step is to set up the heartbeat process. To do this, you need to see what options you have for creating the heartbeat table. Figure [7-4](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Fig4) provides the Command Summary that is used with GGSCI. You will notice that there is a HEARTBEATTABLE object with the commands for ADD, DELETE, ALTER, and INFO listed.



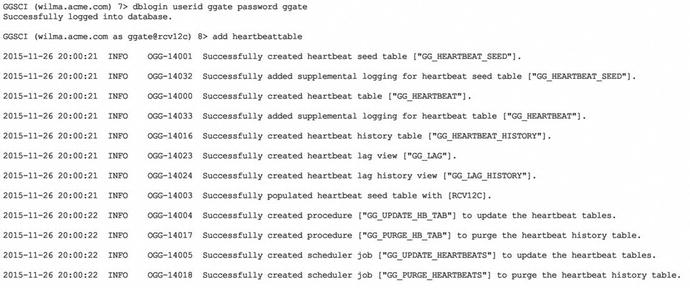
###### Figure 7-4. GGSCI Command Summary

To create the heartbeat table, you simply need to run ADD HEARTBEATTABLE if your heartbeat table name is listed in the GLOBALS file. If not, you will have to provide a schema and table name with the ADD HEARTBEATTABLE command. Listing [7-17](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1) provides an example of the command structure if needed.

###### Listing 7-17. ADD HEARTBEATTABLE Without GLOBALS

GGSCI> dblogin userid [ gg user ] password \*\*\*\*\*\*  
GGSCI> ADD HEARTBEATTABLE [ schema ].[ table name ]

After you have run the command for creating the heartbeat table, you will notice that GoldenGate creates all the items needed for the heartbeat process to work. Figure [7-5](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Fig5) provides a view of all the objects created.



###### Figure 7-5. Objects created by ADD HEARTBEATTABLE

If you are having a hard time reading the image in Figure [7-5](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Fig5), here is a list of all the database objects that the command creates for you.

Tables:

<heartbeat\_table>\_SEED (default GG\_HEARTBEAT\_SEED)  
<heartbeat\_table> (default GG\_HEARTBEAT)  
<heartbeat\_table>\_HISTORY (default GG\_HEARTBEAT\_HISTORY)

Views:

GG\_LAG  
GG\_LAG\_HISTORY

Stored Procedures:

GG\_UPDATE\_HB\_TAB  
GG\_PURGE\_HB\_TAB

Scheduler Jobs:

GG\_UPDATE\_HEARTBEATS  
GG\_PURGE\_HEARTBEATS

As you can tell, unlike the manual process of setting up the heartbeat process, Oracle has integrated all the steps and objects required into a single command. This provides a very effective way to use the heartbeat process within the GoldenGate framework.

## Functions

The last advanced feature is how Oracle GoldenGate can be used to transform data while it is in transit. Oracle GoldenGate provides functions that allow data to be tested while in transit from the source system to the target system. These functions are executed on a column basis. Table [7-3](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab3) provides a summary of these functions for quick reference.

###### Table 7-3. Oracle GoldenGate Functions

| **Function** | **Category** |
| --- | --- |
| CASE | Performance testing |
| EVAL | Performance testing |
| IF | Performance testing |
| COLSTAT | Handling missing columns |
| COLTEST | Handling missing columns |
| DATE | Dates |
| DATEDIFF | Dates |
| DATENOW | Dates |
| COMPUTE | Arithmetic calculations |
| NUMBIN | Strings |
| NUMSTR | Strings |
| STRCAT | Strings |
| STRCMP | Strings |
| STREXT | Strings |
| STREQ | Strings |
| STRFIND | Strings |
| STRLEN | Strings |
| STRLTRIM | Strings |
| STRNCAT | Strings |
| STRNCMP | Strings |
| STRNUM | Strings |
| STRRTRIM | Strings |
| STRSUB | Strings |
| STRTRIM | Strings |
| STRUP | Strings |
| VALONEOF | Strings |
| AFTER | Others |
| BEFORE | Others |
| BEFOREAFTER | Others |
| BINARY | Others |
| BINTOHEX | Others |
| GETENV | Others |
| GETVAL | Others |
| HEXTOBIN | Others |
| HIGHVAL | LOWVAL | Others |
| RANGE | Others |
| TOKEN | Others |

As you can tell, there are quite a few functions that can be used against data as it is being shipped. Many of these functions fit into six different categories, and every category can help in identifying what is happening with the data. To illustration how functions work, let’s take a look at the IF function.

### IF Function

The IF function belongs in the performance testing category and is helpful when you want to test data for conditions before the data arrive in the target database. This function operates just like a normal programming IF statement, by returning one of two values based on a defined condition as mapped in the COLMAP statement of the replicat. We take a closer look at this in a moment.

###### Note

The @IF function can be used with other conditional arguments to test one or more exceptions.

To understand how the @IF functions work, the syntax is as follows:

@IF ( condition, value\_if\_non-zero, value\_if-zero)

To use the @IF function, you need to enable the replicat that is applying transactions to evaluate the data and make changes as required based on the values of the function. To do this, the MAP clause of the replicat parameter file needs to be updated. Listing [7-18](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1) provides an example of a replicat parameter file using the @IF function in the MAP statement.

###### Listing 7-18. Replicat Using @IF Function

--CHECKPARAMS  
REPLICAT REP  
SETENV (ORACLE\_HOME="/u01/app/oracle/product/12.1.0/db12cr1")  
SETENV (ORACLE\_SID="oragg")  
USERID ggate, PASSWORD ggate  
ASSUMETARGETDEFS  
DISCARDFILE ./dirrpt/REP.dsc, append, megabytes 50  
WILDCARDRESOLVE IMMEDIATE  
BATCHSQL  
**map SF.ORDERS, target ATL.ORDERS**   
**COLMAP (USEDEFAULTS, PRICE = @IF(PRICE>100, PRICE, 1000));**

In Listing [7-18](https://learning.oreilly.com/library/view/pro-oracle-goldengate/9781484211793/A340882_1_En_7_Chapter.html#Tab1), you are saying to check the price column of the data coming in to see if the value is greater than 100. If the value is greater, then round the price to 1,000; otherwise, leave the price as the value being replicated.

Using conditional checking within Oracle GoldenGate, data can be evaluated and changed as it is replicated between environments. Doing these conditional checks during replication enables the administrator to quickly make changes to data as needed without spending a lot of time scrubbing data beforehand.