

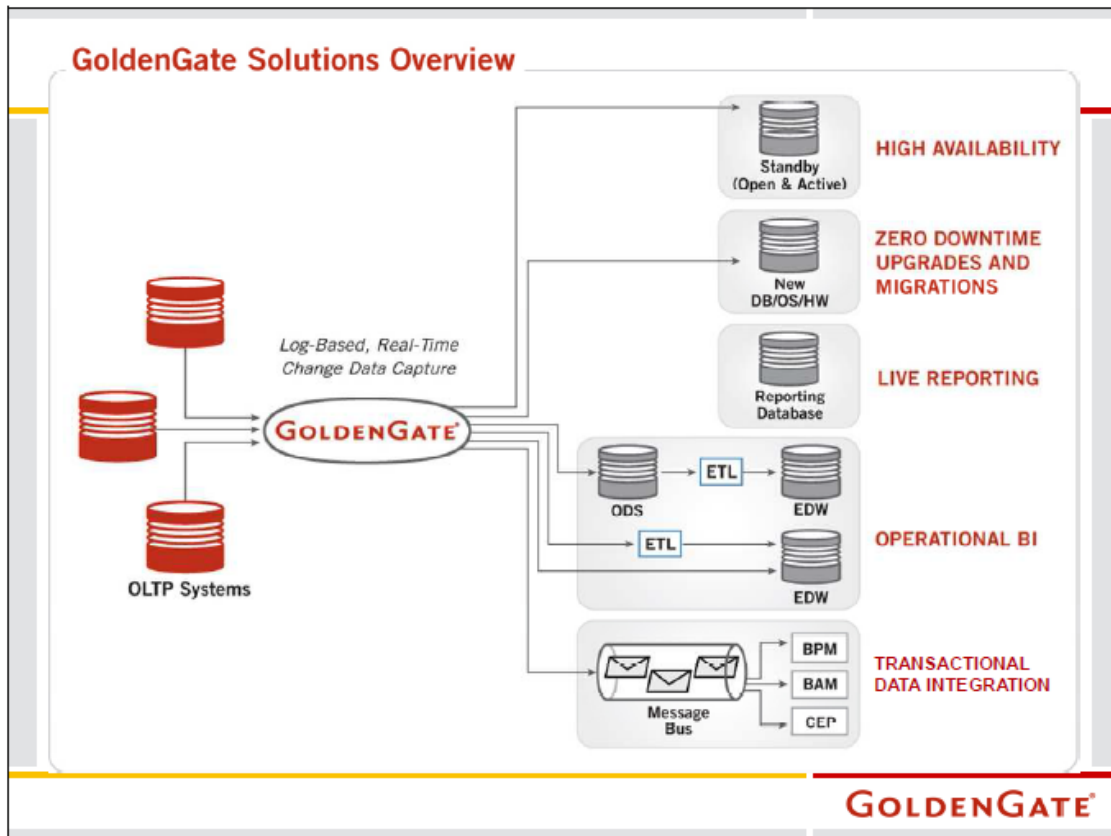
# Golden Gate Study Guide

---

- 1. Introduction**
- 2. Golden Gate Architecture**
- 3. Installing Golden Gate**
- 4. Configuring Golden Gate Environment**
- 5. Golden Gate Extract and Data pump process**
- 6. Golden Gate Replicat process**
- 7. Trail files and Report files**
- 8. Configuring the initial load**
- 9. Data selection, filtering and transformation**
- 10. Bi-directional Replication**
- 11. DDL Replication**
- 12. Configuration Options**

# 1. Introduction

## 1.1 Golden Gate Solution overview



Oracle GoldenGate provides the following data replication solutions:

- High Availability

Live Standby for an immediate fail-over solution that can later re-synchronize with your primary source.

Active-Active solutions for continuous availability and transaction load distribution between two or more active systems.

- Zero-Downtime Upgrades and Migrations

Eliminate downtime for upgrades and migrations.

- Live Reporting

Feeding a reporting database so that you don't burden your source production systems.

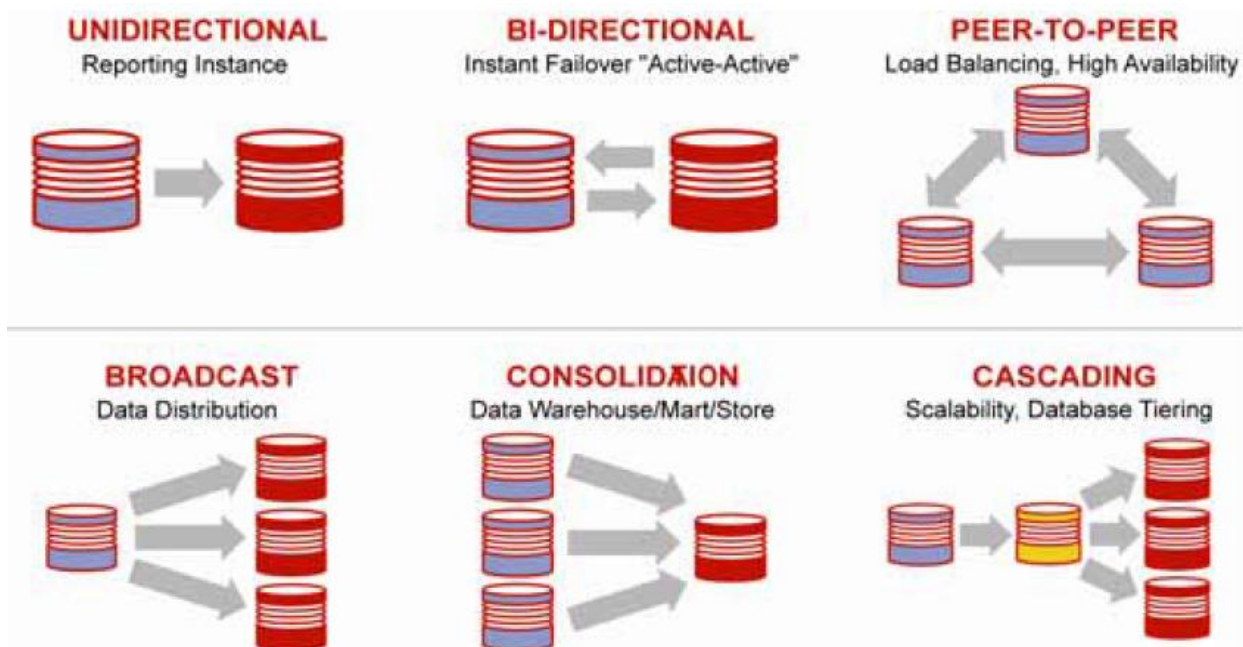
- Operational Business Intelligence (BI)

Real-time data feeds to operational data stores or data warehouses, directly or via ETL tools.

- Transactional data integration

Real-time data feeds to messaging systems for business activity monitoring (BAM), business process monitoring (BPM) and complex event processing (CEP). Uses event-driven architecture (EDA) and service-oriented architecture (SOA).

## 1.2 Golden Gate Supported Topologies:



## 1.3 Supported Databases for Golden Gate

For Capture:

Oracle, MySQL, SQL Server, Sybase, Teradata, DB2 for z/OS, Linux, UNIX, Windows, SQL/MX

For Replication:

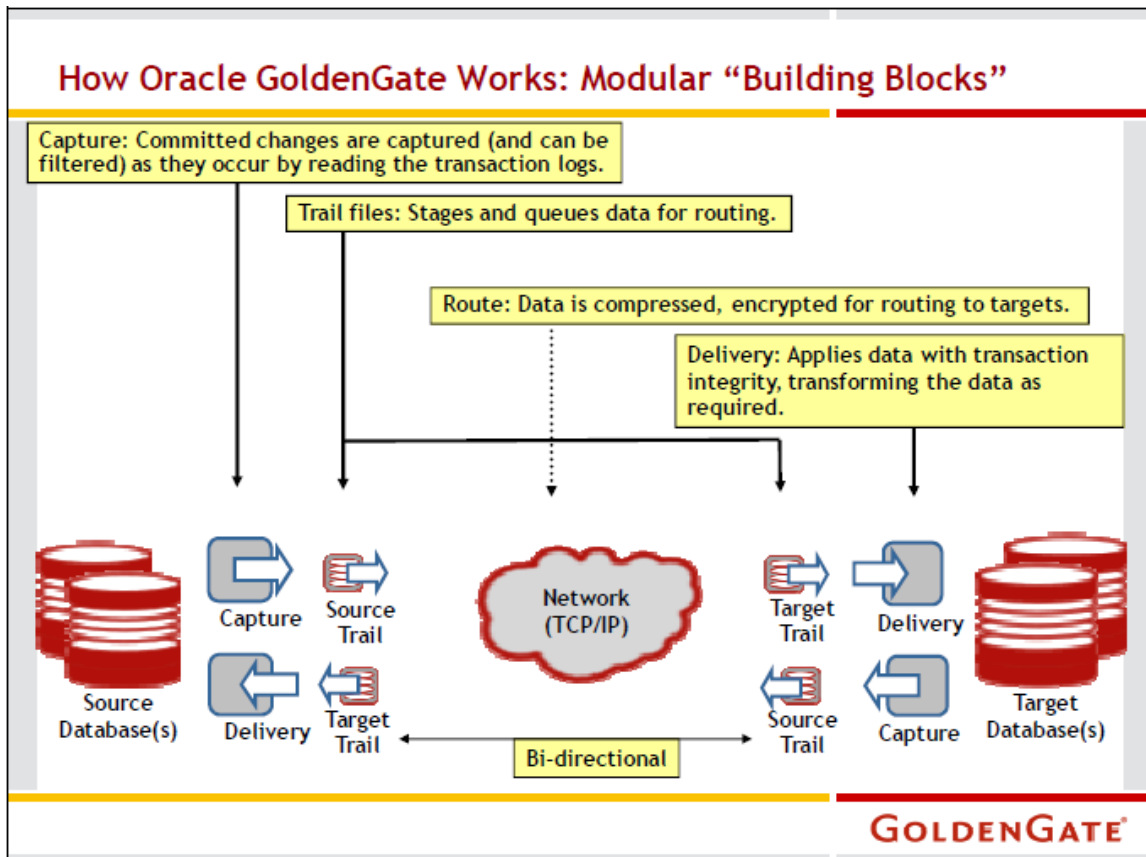
Oracle, MySQL, SQL Server, Sybase, Teradata, DB2 for z/OS, Linux, UNIX, Windows, SQL/MX

, DB2 for i and Times Ten. Other than RDBMS, Golden Gate can deliver data to several data warehouse appliances, ETL servers, and JMS message queues to support Service Oriented Architectures (SOA) and Event-Driven Architectures (EDA).

### ***Golden Gate's Database Support:***

Database	Log-Based Extraction (capture)	Non-Log-Based Extraction** (capture)	Replication (delivery)
DB2 for i*			X
DB2 for Linux, UNIX, Windows	X		X
DB2 for z/OS	X		X
Oracle	X		X
MySQL	X		X
SQL/MX	X		X
SQL Server	X		X
Sybase	X		X
Teradata		X	X
TimesTen*			X

## 1.4 How Does Golden Gate Work



Oracle GoldenGate consists of decoupled modules that are combined to create the best possible solution for your business requirements.

On the source system(s):

- Oracle GoldenGate’s Capture (Extract) process reads data transactions as they occur, by reading the native transaction log, typically the redo log. Oracle GoldenGate only moves changed, committed transactional data, which is only a % of all transactions – therefore operating with extremely high performance and very low impact on the data infrastructure.
- Filtering can be performed at the source or target - at table, column and/or row level.
- Transformations can be applied at the capture or delivery stages.
- Advanced queuing (trail files):

To move transactional data efficiently and accurately across systems, Oracle GoldenGate converts the captured data into a Oracle GoldenGate data format in “trail” files. With both source and target trail files, Oracle GoldenGate’s unique architecture eliminates any single

point of failure and ensures data integrity is maintained – even in the event of a system error or outage.

Routing:

- Data is sent via TCP/IP to the target systems. Data compression and encryption are supported. Thousands of transactions can be moved per second, without distance limitations.

On the target system(s):

- A Server Collector process (not shown) reassembles the transactional data into a target trail.
- The Delivery (Replicat) process applies transactional data to the designated target systems using native SQL calls.

Bi-directional:

- In bi-directional configurations/solutions, this process runs the same in reverse, to concurrently synchronize data between the source and target systems.

## 1.5 Oracle Golden Gate advantages

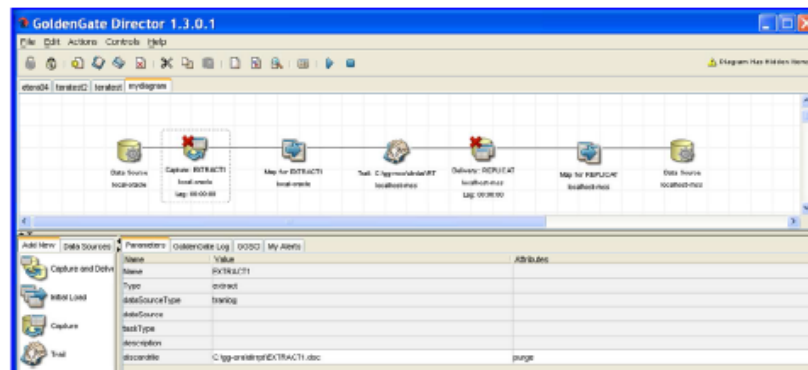
Oracle GoldenGate Advantages		
<b>Movement</b> <ul style="list-style-type: none"><li>▪ Speed<ul style="list-style-type: none"><li>- Subsecond Latency</li></ul></li><li>▪ Volume<ul style="list-style-type: none"><li>- Thousands of TPS</li></ul></li><li>▪ Log-based Capture</li><li>▪ Native, Local Apply</li><li>▪ Efficient IO and Bandwidth Usage</li><li>▪ Bidirectional</li><li>▪ Group Transactions</li><li>▪ Bulk Operations</li><li>▪ Compression</li><li>▪ One-to-Many, Many-to-One</li><li>▪ Cascade</li></ul>	<b>Management</b> <ul style="list-style-type: none"><li>▪ Transaction Integrity</li><li>▪ Transparent Capture</li><li>▪ Guaranteed Delivery</li><li>▪ Conflict Detection, Resolution</li><li>▪ Dynamic Rollback</li><li>▪ Incremental TDM</li><li>▪ Initial Data Load</li><li>▪ GUI-based Monitoring and Configuration</li><li>▪ Proactive Alerts</li><li>▪ Encryption</li><li>▪ Real-Time Deferred or Batch</li><li>▪ Event Markers</li></ul>	<b>Integration</b> <ul style="list-style-type: none"><li>▪ Heterogeneous Data Sources</li><li>▪ Mapping</li><li>▪ Transformation</li><li>▪ Enrichment</li><li>▪ Decoupled Architecture</li><li>▪ Table, Row, Column Filtering</li><li>▪ XML, ASCII, SQL Formats</li><li>▪ Queue Interface</li><li>▪ Stored Procedures</li><li>▪ User Exits</li><li>▪ ETL Integration</li><li>▪ Java/JMS Integration</li></ul>

**GOLDENGATE**

## 1.6 Oracle Golden Gate Director

### Oracle GoldenGate Director

- Manages, defines, configures, and reports on Oracle GoldenGate components
- Key features:
  - Centralized management of Oracle GoldenGate modules
  - Rich-client and Web-based interfaces
  - Alert notifications and integration with 3rd-party monitoring products
  - Real-time feedback
  - Zero-impact implementation



**GOLDENGATE**

Oracle GoldenGate Director is a centralized server-based graphical enterprise application that offers an intuitive way to define, configure, manage, and report on Oracle GoldenGate processes.

Oracle GoldenGate Director is a value added module to centralize management and improve productivity.

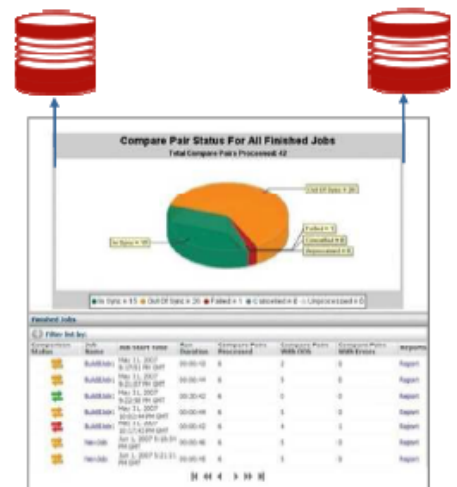
Oracle GoldenGate Director supports all platforms and databases supported by Oracle GoldenGate.



## 1.7 Oracle Golden Gate Veridata

### Oracle GoldenGate Veridata

- A high-speed, low impact data comparison solution
  - Identifies and reports data discrepancies between two databases without interrupting those systems or the business processes they support
  - Supports Oracle, Teradata, SQL Server, NonStop SQL/MP and Enscribe
  - Supports homogeneous and heterogeneous compares
- Benefits:
  - Reduce financial/legal risk exposure
  - Speed and simplify IT work in comparing data sources
  - No disruption to business systems
  - Improved failover to backup systems
  - Confident decision-making and reporting



The screenshot displays the 'Compare Pair Status For All Finished Jobs' interface. At the top, it shows 'Total Compare Pairs Processed: 42'. Below this is a pie chart with three segments: 'In Sync (30)', 'Discrepancy (12)', and 'Comparison Error (0)'. Below the chart is a table with columns: 'Job Name', 'Job Status', 'Job Start Time', 'Job End Time', 'Job Duration', 'Job Progress', 'Job Errors', and 'Job Status'. The table lists several jobs, all with a status of 'Support'.

**GOLDENGATE**

Oracle GoldenGate Veridata is a high-speed data comparison solution that identifies and reports data discrepancies between databases without interrupting ongoing business processes. Using Oracle GoldenGate Veridata, companies can audit and verify large volumes of data across a variety of business applications with certainty, and maintain reliable data synchronization. Oracle GoldenGate Veridata reduces the amount of time and resources required to compare data, it minimizes the impact of human errors, and it ensures that potential problems can be instantly identified and addressed.

### Key Veridata Features:

- Compares large data volumes with high speed and efficiency
- Allows both data sources to be online by handling in-flight transactions
- Performs selective, parallel comparison
- Offers intuitive Web interface and personalized views
- Enables the comparison of databases that are different database versions or on different operating systems
- (HP Nonstop only) supports the comparison of only the data changed since the initial comparison (delta processing).



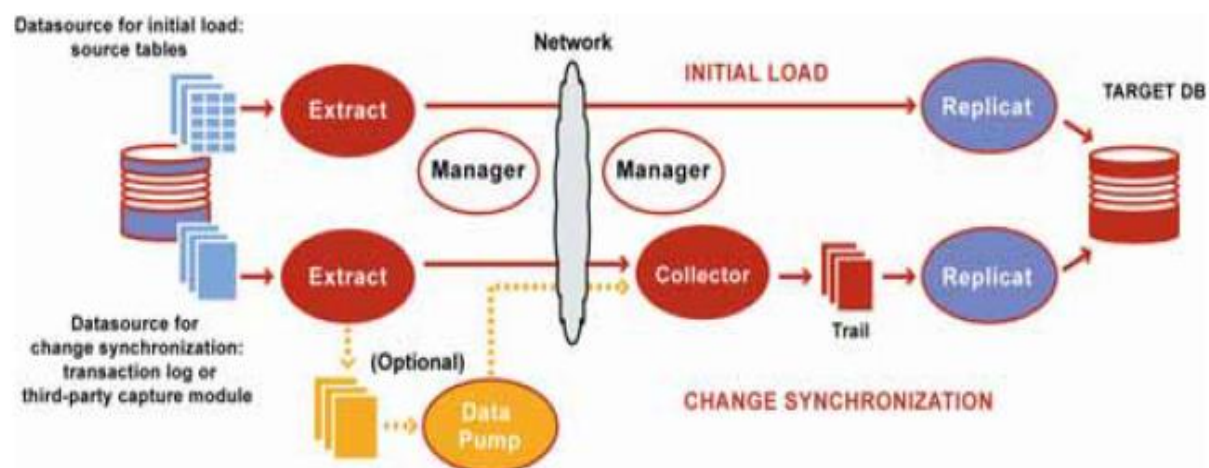
## 2. Architecture

Golden Gate works on changed data capture mechanism which propagates any changes made to the source system.

Golden Gate has following processes and components

- Manager processes
- Collector processes
- Extract processes
- Data pump processes (Optional)
- Replicat processes
- Trail files
- Checkpoint files
- Report/Error files

### 2.1 Golden Gate Logical Architecture



Golden Gate can be configured in two configurations

### 1. Change Synchronization:

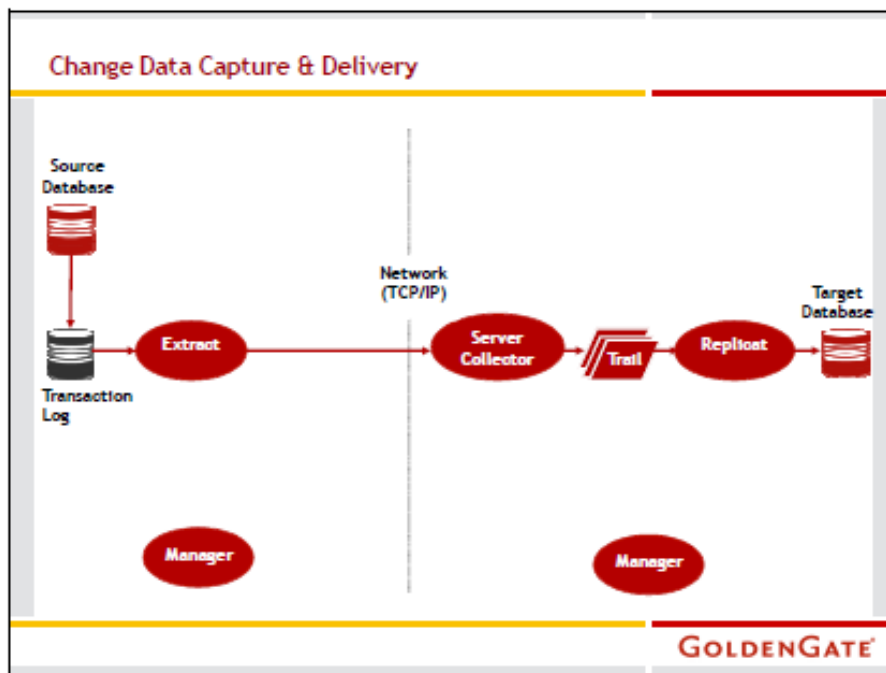
To keep source data synchronized with another set of data, Extract captures DML and DDL operations after the initial synchronization has taken place.

### 2. Initial Load:

For initial data loads, Extract extracts (captures) a current, static set of data directly from their source objects.

## 2.2 Golden Gate Topologies for change synchronization

### 2.2.1 Change Data Capture and Delivery



On the source system:

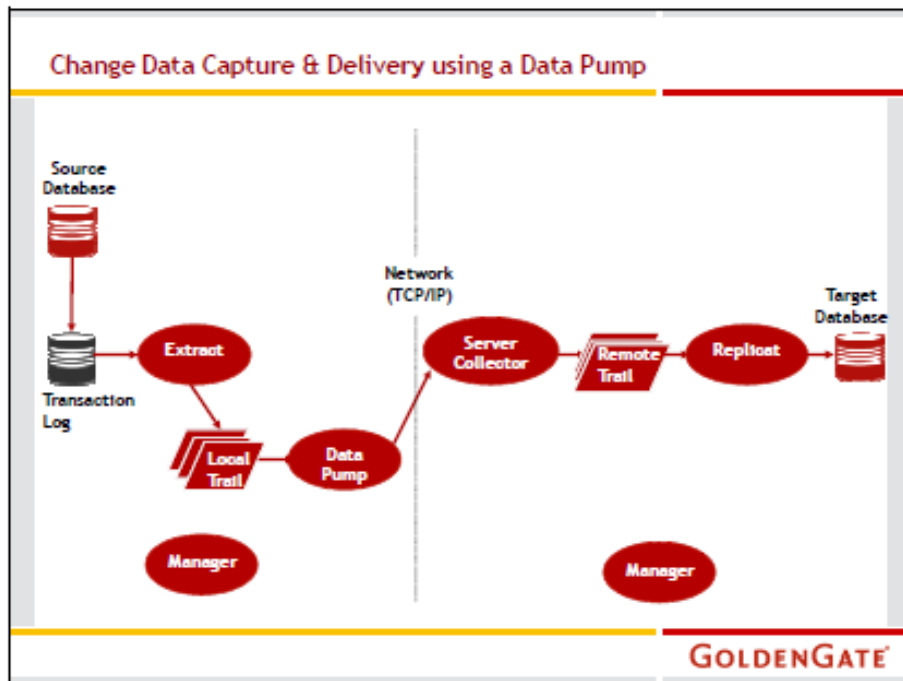
- An *Extract* process captures transactional changes from transaction logs
- The Extract process sends data across a TCP/IP network to the target system.

On the target system:

- A *Server Collector* process reassembles and writes the data to a GoldenGate trail.
- A *Replicat* process reads the trail and applies it to the target database. This can be concurrent with the data capture or performed later.

*Manager* processes on both systems control activities such as starting, monitoring and restarting processes; allocating data storage; and reporting errors and events.

### 2.2.2 Change Data Capture and Delivery using a Data Pump



On the source system:

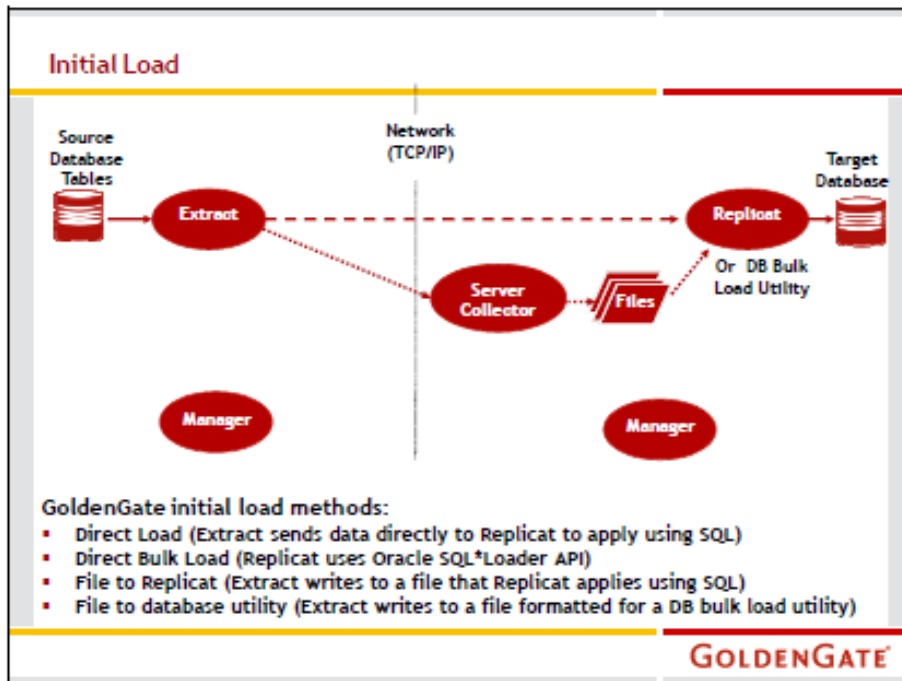
- An *Extract* process captures transactional changes from the database transaction log
- The *Extract* process writes the data to a *local GoldenGate trail*. This preserves the captured data if the network or target trail fails.
- A second *Extract* process (called a *Data Pump*) sends the data across the network to the target system.

On the target system:

- A *Server Collector* process reassembles and writes the data to a GoldenGate trail.
- A *Replicat* process reads the trail and applies it to the target database. This can be concurrent with the data capture or performed later.

*Manager* processes on both systems control activities such as starting, monitoring and restarting processes; allocating data storage; and reporting errors and events.

## 2.3 Golden Gate Topology for Initial Load



Golden Gate Initial Load configuration is generally used to level the data in source and target systems before the start of change synchronization mechanism.

On the source system:

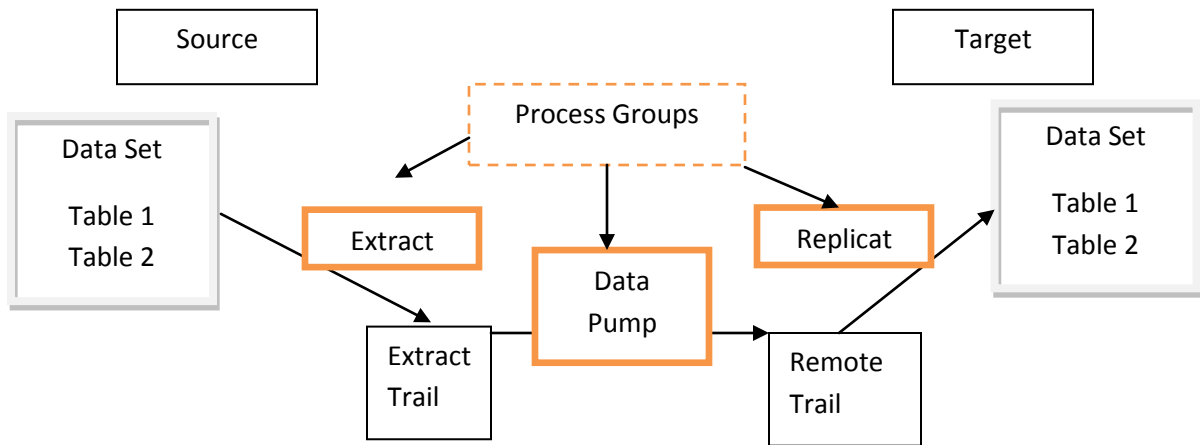
- An *Extract* process captures source data directly from tables
- The Extract process sends data in large blocks across a TCP/IP network to the target system.

On the target system, one of the following scenarios:

1. **Direct Load.** Replicat reads the data stream and concurrently applies the data to the target database using SQL.
2. **Direct Bulk Load (Oracle).** Replicat can apply the data using the Oracle SQL\*Loader API to improve performance.
3. **File to Replicat.** Server Collector reassembles and writes the data to Extract files. Replicat applies the data to the target database using SQL.
4. **File to database utility.** Server Collector reassembles and writes the data to files formatted for a bulk loader, which applies the data to the target database.

*Manager* processes on both systems control activities such as starting, monitoring and restarting processes; allocating data storage; and reporting errors and events.

## 2.4 Golden Gate Process-Groups



We should create separate process groups for logically inter-related data sets.

## 2.5 Golden Gate Checkpoint Mechanism

Golden Gate has its own checkpoint mechanism to handle any outages during the replication process.

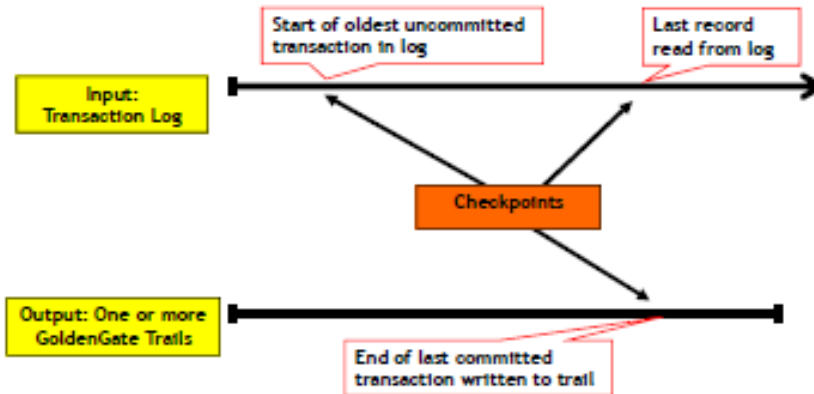
*Checkpoints* store the current read and write positions of a process to disk for recovery purposes. Checkpoints ensure that data changes that are marked for synchronization actually are captured by Extract and applied to the target by Replicat, and they prevent redundant processing. They provide fault tolerance by preventing the loss of data should the system, the network, or an Oracle GoldenGate process need to be restarted. For complex synchronization configurations, checkpoints enable multiple Extract or Replicat processes to read from the same set of trails.

Checkpoints work with inter-process acknowledgments to prevent messages from being lost in the network. Oracle GoldenGate has a proprietary guaranteed-message delivery technology.

### 2.5.1 Extract Checkpointing

## Checkpointing - Extract

- For change data capture, Extract and Replicat save checkpoints to a checkpoint file so they can recover in case of failure
- Extract maintains:
  - 2 input checkpoints
  - 1 output checkpoint for each trail it writes to



GOLDENGATE

Extract creates checkpoints for its positions in the data source and in the trail. Because Extract only captures committed transactions, it must keep track of operations in all open transactions, in the event that any of them are committed. This requires Extract to record a checkpoint where it is currently reading in a transaction log, plus the position of the start of the oldest open transaction, which can be in the current or any preceding log.

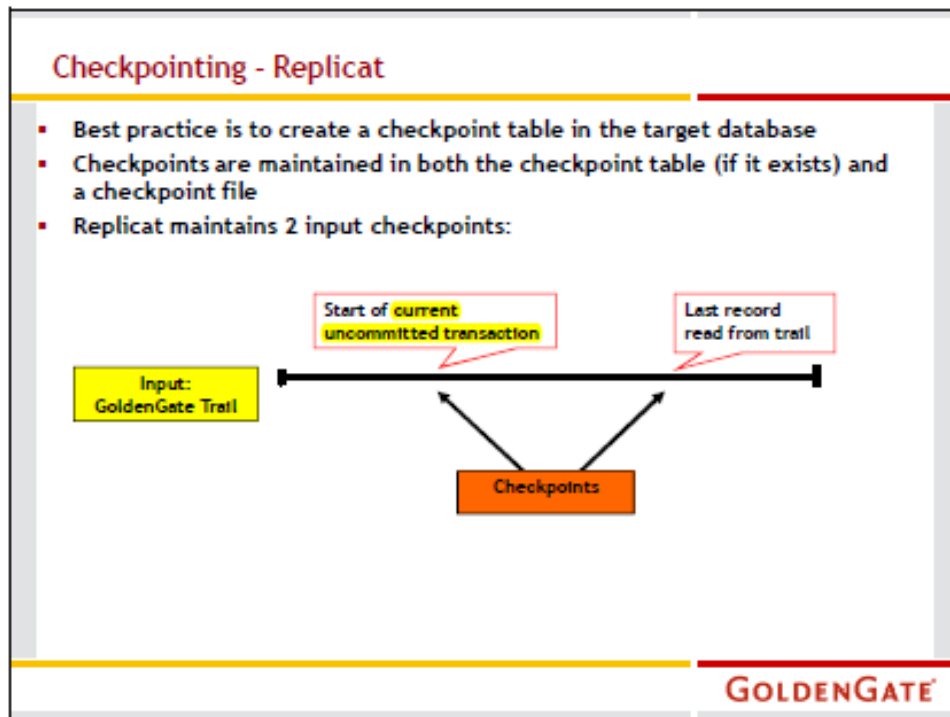
To control the amount of transaction log that must be re-processed after an outage, Extract persists the current state and data of processing to disk at specific intervals, including the state and data (if any) of long-running transactions. If Extract stops after one of these intervals, it can recover from a position within the previous interval or at the last checkpoint, instead of having to return to the log position where the oldest open long running transaction first appeared.

Checkpoints for *End of Last Committed transaction written to trail* and *Start of Oldest Uncommitted transaction (A) in log* are event based checkpoints, which are recorded on commit and start of a transaction.

Whereas, checkpoint for *Last read position* is recorded periodically to control the amount of transaction log that must be re-processed after an outage.



## 2.5.2 Replicat Checkpointing



Replicat creates checkpoints for its position in the trail. Replicat stores its checkpoints in a checkpoint table in the target database to couple the commit of its transaction with its position in the trail file. **The checkpoint table guarantees consistency after a database recovery by ensuring that a transaction will only be applied once, even if there is a failure of the Replicat process or the database process.** For reporting purposes, Replicat also has a checkpoint file on disk in the dirchk sub-directory of the Oracle GoldenGate directory.

Checkpoints are not required for non-continuous types of configurations that can be re-run from a start point if needed, such as initial loads.

## 3. Golden Gate Installation

This guide discusses Golden Gate installation only for Oracle Database. To install golden Gate for other databases and data sources, kindly refer to respective documentation on OTN.

### **3.1 Downloading Oracle GoldenGate**

1. Navigate to <http://edelivery.oracle.com>.
2. On the Welcome page:
  - Select your language.
  - Click Continue.
3. On the Export Validation page:
  - Enter your identification information.
  - Accept the Trial License Agreement (even if you have a permanent license).
  - Accept the Export Restrictions.
  - Click Continue.
4. On the Media Pack Search page:
  - Select the Oracle Fusion Middleware Product Pack.
  - Select the platform on which you will be installing the software.
  - Click Go.
5. In the Results List:
  - Select the Media Pack that you want to download.
  - Click Continue.
6. On the Download page:
  - Click Download for each component that you want. Follow the automatic download process to transfer the mediapack.zip file to your system.

## 3.2 Set ORACLE\_HOME and ORACLE\_SID

### 3.2.1 On Unix based systems

- For single Oracle Instance

Set ORACLE\_HOME and ORACLE\_SID at the system level.

- For multiple Oracle Instances

Set SETENV statement in the Golden Gate Parameter file for each extract and replicat, e.g.

```
SETENV (ORACLE_HOME = "<path to Oracle home location>")  
SETENV (ORACLE_SID = "<SID>")
```

```
EXTRACT xtst01  
SETENV (ORACLE_HOME = "/home/oracle/ora9/product")  
SETENV (ORACLE_SID = "ora9a")  
USERID ggs, PASSWORD ggs  
RMTHOST sysb  
RMTTRAIL /home/ggs/dirdat/rt  
TABLE hr.emp;  
TABLE hr.salary;
```

### 3.2.2 On windows system

- If there is one instance of Oracle on the system, the Registry settings for ORACLE\_HOME and ORACLE\_SID should be sufficient for Oracle GoldenGate. If those settings are incorrect in the Registry and cannot be changed, you can set an override as follows.
  - On the desktop or Start menu (depending on the Windows version), right-click My Computer, and then select Properties.
  - In Properties, click the Advanced tab.
  - Click Environment Variables.
  - Under System Variables, click New.
  - For Variable Name, type ORACLE\_HOME.
  - For Variable Value, type the path to the Oracle binaries.
  - Click OK.
  - Click New again.
  - For Variable Name, type ORACLE\_SID.
  - For Variable Value, type the instance name.
  - Click OK.
- If there are multiple Oracle instances on the system with Extract and Replicat processes connecting to them, do the following.

- Use the preceding procedure (single Oracle instance on system) to set the ORACLE\_HOME and ORACLE\_SID system variables to the first Oracle instance.
- Start all of the Oracle GoldenGate processes that will connect to that instance.
- Repeat the procedure for the next Oracle instance, but first Edit the existing ORACLE\_HOME and ORACLE\_SID variables to specify the new information.
- Start the Oracle GoldenGate processes that will connect to that instance.
- Repeat the Edit and startup procedure for the rest of the Oracle instances.

### **3.3 Setting library paths for dynamic builds on UNIX**

#### ***3.3.1 Set Oracle database shared library path to LD\_LIBRARY\_PATH variable***

Set LD\_LIBRARY\_PATH to point towards Oracle database's shared library directory.

For a Golden Gate process, such as extract or replicat, the following three should always have same bit architecture, either 32 bit or 64 bit or IA64.

- Oracle library versions
- Oracle GoldenGate version
- Database versions

When Oracle Golden gate is connects remotely to the database server through SQL\*Net, additionally the operating system on which the GG instance runs and the operating system on which Database instance runs, must be the same endian.

#### ***3.3.2 Add Golden Gate installation directory to the shared libraries environment variable.***

- (Optional) Add the Oracle GoldenGate installation directory to the PATH environment variable.
- (Required) Add the Oracle GoldenGate installation directory to the shared libraries environment variable.

### UNIX/Linux library path variables per platform

Platform	Environment variable
IBM AIX	LIBPATH
HP-UX	SHLIB_PATH
Sun Solaris HP Tru64 (OSF/1) LINUX	LD_LIBRARY_PATH

**Example** `export LD_LIBRARY_PATH=/ggs/10.0:$LD_LIBRARY_PATH`

## 3.4 Installing Oracle GoldenGate on Linux and UNIX

Follow these steps to install Oracle GoldenGate for Oracle on a Linux or UNIX system or in the appropriate location in a cluster.

### Installing the Oracle GoldenGate files

1. Extract the Oracle GoldenGate mediapack.zip file to the system and directory where you want Oracle GoldenGate to be installed.
2. Run the command shell.
3. Change directories to the new Oracle GoldenGate directory.
4. From the Oracle GoldenGate directory, run the GGSCI program.

```
GGSCI
```

5. In GGSCI, issue the following command to create the Oracle GoldenGate working directories.

```
CREATE SUBDIRS
```

6. Issue the following command to exit GGSCI.

```
EXIT
```

## 3.5 Installing Oracle GoldenGate on Windows

### 3.5.1 *Installing the Oracle GoldenGate files*

1. Unzip the downloaded file(s) by using WinZip or an equivalent compression product.
2. Move the files in binary mode to a folder on the drive where you want to install Oracle GoldenGate. Do not install Oracle GoldenGate into a folder that contains spaces in its name, even if the path is in quotes. For example:

**C:\“Oracle GoldenGate” is not valid.**

**C:\Oracle\_GoldenGate is valid.**

3. From the Oracle GoldenGate folder, run the GGSCI program.
4. In GGSCI, issue the following command to create the Oracle GoldenGate working directories.

**CREATE SUBDIRS**

5. Issue the following command to exit GGSCI.

**EXIT**

### 3.5.2 *Specifying a custom Manager name*

You must specify a custom name for the Manager process if either of the following is true:

- You want to use a name for Manager other than the default of GGSMGR.
- There will be multiple Manager processes running as Windows services on this system.

Each Manager on a system must have a unique name. Before proceeding further, note the names of any local Manager services.

#### **To specify a custom Manager name**

1. From the directory that contains the Manager program, run GGSCI.
2. Issue the following command.

**EDIT PARAMS ./GLOBALS**

**NOTE** The ./ portion of this command must be used, because the GLOBALS file must reside at the root of the Oracle GoldenGate installation file.

3. In the file, add the following line, where <name> is a one-word name for the Manager service.

**MGRSERVNAME <name>**

4. Save the file. The file is saved automatically with the name GLOBALS, without a file extension. Do not move this file. It is used during installation of the Windows service and during data processing.

### **3.5.3 Installing Manager as a Windows service**

#### **To install Manager as a Windows service**

1. (Recommended) Log on as the system administrator.
2. Click Start > Run, and type cmd in the Run dialog box.
3. From the directory that contains the Manager program that you are installing as a service, run the install program with the following syntax:

**install <option> [...]**

**Where: <option> is one of the following:**

<b>Option</b>	<b>Description</b>
<b>ADDEVENTS</b>	<b>Adds Oracle GoldenGate events to the windows Event Manager. By default, Oracle GoldenGate errors are generic. To produce more specific error content, copy the following files from the Oracle GoldenGate installation directory to the SYSTEM32 directory.</b>  <b>category.dll</b>  <b>ggsmg.dll</b>
<b>ADDSERVICE</b>	<b>Adds Manager as a service with the name that is specified with the MGRSERVNAME parameter in the GLOBALS file, if one exists, or by the default of GGSMGR. ADDSERVICE configures the service to run as the Local System account, the standard for most Windows applications because the service can be run independently of user logins and password changes. To run</b>

	Manager as a specific account, use the USER and PASSWORD options
AUTOSTART	Sets the service that is created with ADDSERVICE to start at system boot time. This is the default unless MANUALSTART is used.
MANUALSTART	Sets the service that is created with ADDSERVICE to start manually through GGSCI, a script, or the Services applet of the Control Panel. The default is AUTOSTART.
USER <name>	Specifies a domain user account that executes Manager. For <name>, include the domain name, a backward slash, and the user name, for example HEADQT\GGSMGR.  By default, the Manager service is installed to use the Local System account.
PASSWORD <password>	Specifies the password for the user that is specified with USER.

### 3.6 Golden Gate home – Directory structure

<b>Prepare Environment: Installation - GoldenGate Directories</b>	
<u>Directory</u>	<u>Contents</u>
dirchk	GoldenGate checkpoint files
dirdat	GoldenGate trail and extract files
dirdef	Data definitions produced by DEFGEN and used to translate heterogeneous data
dirpcs	Process status files
dirprm	Parameter files
dirrpt	Process report files
dirsql	SQL scripts
dirtmp	Temporary storage for transactions that exceed allocated memory

**GOLDENGATE**



## dirchk

Contains the checkpoint files created by Extract and Replicat processes, which store current read and write positions to support data accuracy and fault tolerance. Written in internal GoldenGate format. Do not edit these files.

The file name format is <group name><sequence number>.<ext> where <sequence number> is a sequential number appended to aged files and <ext> is either cpe for Extract checkpoint files or cpr for Replicat checkpoint files. Examples: ext1.cpe, rep1.cpr

## dirdat

The default location for GoldenGate trail files and extract files created by Extract processes to store records of extracted data for further processing, either by the Replicat process or another application or utility. Written in internal GoldenGate format. Do not edit these files.

File name format is a user-defined two-character prefix followed by either a six-digit sequence number (trail files) or the user-defined name of the associated Extract process group (extract files). Examples: rt000001, finance

## dirdef

The default location for data definitions files created by the DEFGEN utility to contain source or target data definitions used in a heterogeneous synchronization environment. Written in external ASCII.

File name format is a user-defined name specified in the DEFGEN parameter file. These files may be edited to add definitions for newly created tables. If you are unsure of how to edit a definitions file, contact technical support. Example: defs.dat

## dirpcs

Default location for status files. File name format is <group>.<extension> where <group> is the name of the group and <extension> is either pce (Extract), pcr (Replicat), or pcm (Manager).

These files are only created while a process is running. The file shows the program name, the process name, the port, and process ID that is running. Do not edit these files. Examples: mgr.pcm, ext.pce

## dirprm

The default location for GoldenGate parameter files created by GoldenGate users to store runtime parameters for GoldenGate process groups or utilities. Written in external ASCII format.

File name format is <group name/user-defined name>.prm or mgr.prm. These files may be edited to change GoldenGate parameter values. They can be edited directly from a text editor or by using the EDIT PARAMS command in GGSCI. Examples: defgen.prm, finance.prm

dirrpt

The default location for process report files created by Extract, Replicat, and Manager processes to report statistical information relating to a processing run. Written in external ASCII format.

File name format is <group name><sequence number>.rpt where <sequence number> is a sequential number appended to aged files. Do not edit these files. Examples: fin2.rpt, mgr4.rpt

dirsql

The default location for SQL scripts.

dirtmp

The default location for storing large transactions when the size exceeds the allocated memory size. Do not edit these files.

### **3.7 Golden Gate Documentation**

- Release Notes
- Upgrade Guide
- Administrator's Guide
- Reference Guide
- Troubleshooting and Tuning Guide
- <Database Specific > Installation guide

[http://docs.oracle.com/cd/E35209\\_01/index.htm](http://docs.oracle.com/cd/E35209_01/index.htm)

## 4. Configuring Golden Gate Environment

### 4.1 Preparing the Oracle database for Oracle Golden Gate

#### 4.1.1 Oracle Golden Gate supported Oracle Database Versions

Golden Gate 11.1 and earlier	Golden Gate 11.2 and later
Oracle Database 9.2	Oracle Database 10.2
Oracle Database 10.1	Oracle Database 11.1
	Oracle Database 11.2

#### 4.1.2 Oracle Database user for Golden Gate

Create a source database user and a target database user, each one dedicated to Oracle GoldenGate on the source and target systems. It can be the same user for all of the Oracle GoldenGate processes that must connect to a source or target Oracle database:

- Extract (source database): This user performs metadata queries on the source database, and to fetch data from the source tables for data types that are not directly supported from the redo stream.
- Replicat (target database): This user is used to create the Replicat checkpoint table and to apply DML, DDL, and initial load operations.
- Manager (source database, if using DDL support): This user performs maintenance on the Oracle GoldenGate database objects if DDL support is being used.
- DEFGEN (source or target database): This user performs local metadata queries to build a data-definitions file that supplies the metadata to remote Oracle GoldenGate instances.

## Database privileges by Oracle GoldenGate process

User Privilege	Extract (Source Side)	Replicat (Target Side)
CREATE SESSION, ALTER SESSION	X	X
RESOURCE	X	x
CONNECT	X	X
SELECT ANY DICTIONARY	X	X
FLASHBACK ANY TABLE or FLASHBACK ON <owner.table>	X	
SELECT ANY TABLE or SELECT ON <owner.table>	X	X
INSERT, UPDATE, DELETE ON <target tables>		X
CREATE TABLE		X
EXECUTE on DBMS_FLASHBACK package	X	

In addition, execute the following command in SQL\*Plus as SYSDBA:

```
EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE('GGUSER','*',TRUE);
```

Where *GGUSER* is the database user ID used in GGSCI DBLogin commands.

### 4.1.3 Configuring the source database to log key values

GGSCI provides commands to configure the source database to log the appropriate key values whenever it logs a row change, so that they are available to Oracle GoldenGate in the redo record. By default, the Oracle database only logs column values that are changed. The appropriate command must be issued before you start Oracle GoldenGate processing.

You will enable some or all of the following supplemental logging types:

- Enabling database-level supplemental logging
- Enabling schema-level supplemental logging
- Enabling table-level supplemental logging

## Enabling database-level supplemental logging

Oracle GoldenGate requires enabling database-level supplemental logging.

1. Log in to SQL\*Plus as a user with ALTER SYSTEM privilege, and then issue the following command to enable minimal supplemental logging at the database level. This logging is required to process updates to primary keys and chained rows.

```
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
```

2. To start the supplemental logging, switch the log files.

```
ALTER SYSTEM SWITCH LOGFILE;
```

## Enabling schema-level supplemental logging

To issue ADD SCHEMATRANDATA

1. Apply Oracle Patch 10423000 to the source Oracle database if the version is earlier than 11.2.0.2.
2. Run GGSCI on the source system.
3. Issue the DBLOGIN command as a user that has privilege to enable schema-level supplemental logging.

```
DBLOGIN USERID <user>, PASSWORD <password> [<encryption options>]
```

4. Issue the following command for each schema for which you want to capture data changes.

```
ADD SCHEMATRANDATA <schema>
```

As an example, the following commands enable supplemental logging for the FINANCE and HR schemas.

```
ADD SCHEMATRANDATA FINANCE
```

```
ADD SCHEMATRANDATA HR
```

## Enabling table-level supplemental logging

The ADD TRANDATA command is required to enable table-level supplemental logging of key values for use by Oracle GoldenGate. You can also use the COLS option of this command to log non-key columns for use in a KEYCOLS clause or for filtering or manipulation.

To issue ADD TRANDATA

1. Run GGSCI on the source system.
2. Issue the DBLOGIN command as a user that has privilege to enable table-level supplemental logging.

```
DBLOGIN USERID <user>, PASSWORD <password> [<encryption options>]
```

3. Issue the following command.

```
ADD TRANDATA <table> [, COLS <columns>] [, NOKEY]
```

Where:

- <table> is the owner and name of the table. You can use a wildcard for the table name, but not the owner name.
  - COLS <columns> logs non-key columns that are required for a KEYCOLS clause or for filtering and manipulation.
  - NOKEY prevents the logging of the primary key or unique key. Requires a KEYCOLS clause in the TABLE and MAP parameters and a COLS clause in the ADD TRANDATA command to log the KEYCOLS columns.
4. If using ADD TRANDATA with the COLS option, create a unique index for those columns on the target to optimize row retrieval. If you are logging those columns as a substitute key for a KEYCOLS clause, make a note to add the KEYCOLS clause to the TABLE and MAP statements when you configure the Oracle GoldenGate processes.

### ***4.1.4 Enable archive logging as a secondary data source in case the online logs recycle before Extract is finished with them.***

## 4.2 Creating the Oracle GoldenGate instance

Each Oracle GoldenGate installation is rooted in the Manager process. This is the controller process that instantiates the Oracle GoldenGate processes, allocates port numbers, and performs file maintenance. Together, the Manager process and its child processes, and their related programs and files comprise an Oracle GoldenGate instance.

To run Oracle GoldenGate, a Manager process must be running on all systems that will be part of the Oracle GoldenGate environment. To run Manager, you first create a parameter file for it.

### 4.2.1 Golden Gate Manager

The Manager provides a command-line interface to perform a variety of tasks:

- Starting, stopping, and monitoring Oracle GoldenGate processes
  - Setting parameters to configure Oracle GoldenGate processes
  - Error and lag reporting
  - Resource management
  - Trail file management
- The Manager process must be running on each system before Extract or Replicat can be started.
  - Manager parameters are entered in the **mgr.prm** file under the **dirprm** directory.

#### To create the Manager parameter file

1. From the Oracle GoldenGate directory, run the ggsci program to open the Oracle GoldenGate Software Command Interface (GGSCI).
2. In GGSCI, edit the Manager parameter file.

```
EDIT PARAMS MGR
```

3. Add the Manager parameters, each on one line. If a parameter statement must span multiple lines, use an ampersand (&) before each line break.
  - The only required Manager parameter is PORT, but DYNAMICPORTLIST is strongly recommended.

- In a cluster environment, configure Manager with the AUTOSTART and AUTORESTART parameters, so that the Oracle GoldenGate processes start or restart automatically when Manager is started or fails over.
- Use PURGEOLDEXTRACTS to manage the accumulation of trail files.

4. Save, and then close the file.

#### ***4.2.2 Creating Source definitions for heterogeneous environments using defgen***

- When capturing, transforming, and delivering data across disparate systems and databases, Oracle GoldenGate must understand both the source and target layouts.
- The **defgen** utility produces a file containing a definition of the layouts of the source files and tables.
- This source definition file is used to interpret layouts for data stored in Oracle GoldenGate trails.
- At startup, Replicat reads the definition file specified with the **SourceDefs** parameter.
- defgen is initiated from the command prompt:

```
defgen paramfile <paramfile> [ reportfile <reportfile> ]
```

- Definitions are saved to the file specified in the parameter file. This file must be transferred to the target system as a text file.



### Prepare Environment: Source Definitions - Sample DEFGEN Parameters

```
DEFSFILE /ggs/dirdef/source.def, PURGE
SOURCEDB mydb, USERID ggs, PASSWORD ggs
TABLE SALES.ACCOUNT;
TABLE SALES.PRODUCT;
```

<u>Parameter</u>	<u>Specifies</u>
DEFSFILE	The output definitions file location and name
SOURCEDB	The database name (if needed)
USERID	The user ID and password (if needed) to access the database
TABLE	The table(s) to be defined

**GOLDENGATE**

#### 4.2.3 GLOBALS Parameters

- Control things common to all processes in a GoldenGate instance
- Can be overridden by parameters at the process level
- Must be created before any processes are started
- Stored in <GoldenGate install directory>/GLOBALS
- (GLOBALS is uppercase, no extension)
- Must exit GGSCI to save
- Once set, rarely changed
- Parameters most commonly used

**MGRSERVNAME ggsmanager1**

Defines a unique Manager service name on Windows systems

**CHECKPOINTTABLE dbo.ggschkpt**

Defines the table name used for Replicat's checkpoint table

## 5. Golden Gate Extract and Data Pump Process

### 5.1 Configuring Change Capture (Primary Extract)

#### 5.1.1 Primary Extract Overview

Extract can be configured to:

- Capture changed data from database logs
- Distribute data from local trails to remote systems (data pump)
- Capture data directly from source tables for initial data load

#### 5.1.2 How to configure Change Capture

On the source system:

- Add a primary Extract (reading from source transaction logs) with an associated parameter file
- Optionally, add a local trail and a data pump Extract (reading from the local trail) with an associated parameter file
- Add a remote trail
- Start the Extract(s)

Golden Gate Extract process for Change Data Capture (CDC) captures all the changes that are made to the objects that you configure for synchronization.

To configure Extract to capture changes from transaction logs, perform the following steps:

- Set up a parameter file for Extract with the **GGSCI EDIT PARAMS** command.
- Set up an initial Extract checkpoint into the logs with the **GGSCI ADD EXTRACT** command.
- Optionally, create a local trail using the **GGSCI ADD EXTTRAIL** command and a data pump Extract (and parameter file) reading from the local trail.
- Set up a remote trail using the **GGSCI ADD RMTTRAIL** command.

- Start the Server Collector process on the target system or let the Manager start the Server Collector dynamically.
- Start Extract using the GGSCI START EXTRACT command. For example:  
**GGSCI> START EXTRACT FINANCE**
- GGSCI sends this request to the Manager process, which in turn starts Extract.

Example:

Add the initial Extract checkpoint with the GGSCI command ADD EXTRACT:

**ADD EXTRACT <group name>, <data source>, <starting point> [, <processing options>]**

#### 5.1.2.1 ADD EXTRACT <data source>

<data source>	Source (and when used)
SOURCEISTABLE	Database table (initial data load)
TRANLOG [<bsds name>]	Transaction log (change capture) [DB2 z/OS]
EXTFILESOURCE <file name>	Extract file (data pump)
EXTTRAILSOURCE <trail name>	Trail (data pump)

#### SOURCEISTABLE

Creates an Extract task that extracts entire records from the database for an initial load. If SOURCEISTABLE is not specified, ADD EXTRACT creates an online change-synchronization process, and one of the other data source options must be specified. When using SOURCEISTABLE, do not specify service options. Task parameters must be specified in the parameter file.

### **TRANLOG [<bsds name>]**

Specifies the transaction log as the data source. Use this option for log-based extraction. TRANLOG requires the BEGIN option. Use the <bsds name> option for DB2 on a z/OS system to specify the BSDS

(Bootstrap Data Set) file name of the transaction log. Make certain that the BSDS name you provide is the one for the DB2 instance to which the Extract process is connected. GoldenGate does not perform any validations of the BSDS specification.

### **EXTFILESOURCE <file name>**

Specifies an extract file as the data source. Use this option with a secondary Extract group (data pump) that acts as an intermediary between a primary Extract group and the target system. For <file name>, specify the fully qualified path name of the file, for example c:\ggs\dir\dat\extfile.

### **EXTTRAILSOURCE <trail name>**

Specifies a trail as the data source. Use this option with a secondary Extract group (data pump) that acts as an intermediary between a primary Extract group and the target system. For <trail name>, specify the fully qualified path name of the trail, for example c:\ggs\dir\dat\aa.

#### **5.1.2.2 ADD EXTRACT <starting Point>**

<u>&lt;starting point&gt;</u>	<u>Database</u>
BEGIN {NOW   <datetime> }	Any
EXTSEQNO <seqno>, EXTRBA <relative byte address>	Oracle, SQL/MX
EXTRBA <relative byte address>	DB2 z/OS
EOF   LSN <value>	DB2 LUW
LSN <value>	SQL Server, Ingres
LOGNUM <log number>, LOGPOS <byte offset>	c-tree
PAGE <data page>, ROW <row>	Sybase

**The starting point is indicated by one of the following:**

**BEGIN** specifies when the Extract begins processing.

- For all databases except DB2 LUW, NOW specifies the time at which the ADD EXTRACT command is issued.

- For DB2 LUW, NOW specifies the time at which START EXTRACT takes effect. <datetime> specifies the start date and time in the format: yyyy-mm-dd [hh:mi:[ss[.cccc]]].

Several parameters specify the position with the log or trail to begin processing:

**EXTSEQNO <seqno>, EXTRBA <relative byte address>**

Valid for a primary Extract for Oracle and NonStop SQL/MX, and for a data pump Extract.

Specifies one of the following:

- sequence number of an Oracle redo log and RBA within that log at which to begin capturing data.

- the NonStop SQL/MX TMF audit trail sequence number and relative byte address within that file at which to begin capturing data. Together these specify the location in the TMF Master Audit Trail (MAT).

- the file in a trail in which to begin capturing data (for a data pump). Specify the sequence number, but not any zeroes used for padding. For example, if the trail file is c:\ggs\dir\dat\aa000026, you would specify EXTSEQNO 26. By default, processing begins at the beginning of a trail unless this option is used.

**EXTRBA <relative byte address>**

Valid for DB2 on z/OS. Specifies the relative byte address within a transaction log at which to begin capturing data.

**EOF | LSN <value>**

Valid for DB2 LUW. Specifies a start position in the transaction logs when Extract starts.

EOF configures processing to start at the active LSN in the log files. The active LSN is the position at the end of the log files that the next record will be written to. Any active transactions will not be captured.

LSN <value> configures processing to start at an exact LSN if a valid log record exists there. If one does not exist, Extract will abend. Note that, although Extract might position to a given LSN, that LSN might not necessarily be the first one that Extract will process. There are

numerous record types in the log files that Extract ignores, such as DB2 internal log records. Extract will report the actual starting LSN to the Extract report file.

**LSN <value>**

Valid for SQL Server or Ingres. Specifies the LSN in a SQL Server or Ingres transaction log at which to start capturing data. The LSN specified should exist in a log backup or the online log.

- For SQL Server, an LSN is composed of three hexadecimal numbers separated by colons. The first is the virtual log file number, the second is the segment number within the virtual log, and the third is the entry number.

- For Ingres, an LSN is two, 4-byte unsigned integers, separated by a colon. For example, to specify an LSN of 1206396546,43927 (as viewed in an Ingres utility), you would enter 1206396546:43927.

- An alias for this option is EXTLSN.

**LOGNUM <log number>, LOGPOS <byte offset>**

Valid for c-tree. Specifies the location in a c-tree transaction log at which to start capturing data. <log number> is the number of the c-tree log file. <byte offset> is the relative position from the beginning of the file (0 based).

**PAGE <data page>, ROW <row>**

Valid for Sybase. Specifies a data page and row that together define a start position in a Sybase transaction log.

### 5.1.2.3 ADD EXTRACT <processing options>

<u>&lt;processing options&gt;</u>	<u>Specifies</u>
<b>DESC</b> "<description>"	Description of Extract group
<b>THREADS</b> <n>	Number of redo threads when extracting from an Oracle RAC clustered database
<b>PARAMS</b> <file name>	Alternative parameter file name (fully qualified)
<b>PASSTHRU</b>	Used only in Data Pumps. Passes the data through without any transformation.
<b>REPORT</b> <file name>	Alternative report file name (fully qualified)

### 5.1.2.4 ADD EXTRACT Examples

Create an Extract group named "finance" that extracts database changes from the transaction logs. Start extracting with records generated at the time when you add the Extract group.

#### **ADD EXTRACT finance, TRANLOG, BEGIN NOW**

Create an Extract group named "finance" that extracts database changes from the transaction logs. Start extracting with records generated at 8:00 on January 31, 2006.

#### **ADD EXTRACT finance, TRANLOG, BEGIN 2006-01-31 08:00**

Create a data-pump Extract group named "finance" that reads from the GoldenGate trail c:\ggs\dir\dat\lt.

#### **ADD EXTRACT finance, EXTTRAILSOURCE c:\ggs\dir\dat\lt**

Create an initial-load Extract named "load".

#### **ADD EXTRACT load, SOURCEISTABLE**

**TRANLOG** specifies the transaction log as the data source.

**BEGIN NOW** specifies that the change capture is to begin immediately.

**ADD RMTTRAIL** Identifies the GoldenGate trail on the target system and the name of the EXTRACT group.

### 5.1.2.5 Edit Extract Parameters

Create/edit an Extract parameter file with the GGSCI command:

**EDIT PARAMS <group name>**

```
EXTRACT xtst01  
USERID GoldenUser, PASSWORD password  
RMTHOST serverx, MGRPORT 7809  
RMTTRAIL /ggs/dirdat/rt  
TABLE SALES.ORDERS;  
TABLE SALES.INVENTORY;
```

**USERID <login>, PASSWORD <pw>** supplies database credentials. (SOURCEDB <dsn> is not required for Oracle).

**RMTHOST <hostname>** specifies the target system.

**MGRPORT <port>** specifies the port where Manager is running.

**RMTTRAIL <trail id>** specifies the GoldenGate trail on the target system.

**TABLE <table name>** specifies a source table for which activity will be extracted.

### 5.1.2.6 Add a Local/Remote Trail

Add a local or remote trail with the GGSCI command:

```
ADD EXTTRAIL | RMTTRAIL <trail name>  
  
, EXTRACT <group name>  
  
[, MEGABYTES <n>]
```

If using a data pump:

The primary Extract needs a local trail (EXTTRAIL)

The data pump Extract needs a remote trail (RMTTRAIL)

Examples:

```
ADD EXTTRAIL c:\ggs\dirdat\aa, EXTRACT finance, MEGABYTES 10  
ADD RMTTRAIL c:\ggs\dirdat\bb, EXTRACT parts, MEGABYTES 5
```

**<trail name>** The fully qualified path name of the trail. The actual trail name can contain only two characters. GoldenGate appends this name with a six-digit sequence number whenever a



new file is created. For example, a trail named /ggs/dirdat/tr would have files named /ggs/dirdat/tr000001, /ggs/dirdat/tr000002, and so forth.

**<group name>** The name of the Extract group to which the trail is bound. Only one Extract process can write data to a trail.

**MEGABYTES <n>** The maximum size, in megabytes, of a file in the trail. The default is 10.

### **5.1.3 Extract Parameters**

**Extract parameters specify:**

- Group name – associates with a checkpoint file
- Where to send the data
  - Local system
  - Multiple remote systems
  - One to many GoldenGate trails
- What is being captured
  - Which tables
  - Which rows and columns
  - Which operations
- Which column mapping to apply
- Which data transformations to apply

**All Extract parameters assume a default value:**

- Capture all insert, update and delete operations
  - Committed data only
  - Full image for inserts
  - Only primary key and changed columns for updates
  - Only primary key for deletes
  - Only after-image of update

- Send data without transformation
- Buffer transactions
  - Until a block is full or
  - Until time elapses
- Based on average transaction volumes

### Extract Parameters

<u>Purpose</u>	<u>Examples</u>
General	SETENV, GETENV, OBEY
Processing method	BEGIN, END, PASSTHRU
Database login	SOURCEDB, USERID
Selecting and mapping data	IGNOREINSERTS, GETUPDATEBEFORES, TABLE
Routing data	EXTTRAIL, RMTHOST, RMTTRAIL
Formatting data	FORMATASCII, FORMATSQL, FORMATXML, NOHEADERS
Custom processing	CUSEREXIT, INCLUDE, MACRO, SQLEXEC
Reporting	REPORT, REPORTCOUNT, STATOPTIONS
Error handling	DISCARDFILE, DDLERROR
Tuning	ALLOCFILES, CHECKPOINTSECS, DBOPTIONS
Maintenance	PURGEOLDEXTRACTS, REPORTROLLOVER
Security	ENCRYPTTRAIL, DECRYPTTRAIL

### Extract TRANLOGOPTION parameter

Use the TRANLOGOPTIONS parameter to control database-specific aspects of log-based extraction

Examples: Controlling the archive log

**TRANLOGOPTIONS ALTARCHIVEDLOGFORMAT log\_%t\_%s\_%r.arc**

Specifies an alternative archive log format.

**TRANLOGOPTIONS ALTARCHIVELOGDEST /oradata/archive/log2**

Specifies an alternative archive log location.

**TRANLOGOPTIONS ARCHIVEDLOGONLY**

Causes Extract to read from the archived logs exclusively.

Examples: Loop prevention

**TRANLOGOPTIONS EXCLUDEUSER ggsrep**

Specifies the name of the Replicat database user so that those transactions are not captured by Extract.

**TRANLOGOPTIONS EXCLUDETRANS "ggs\_repl"**

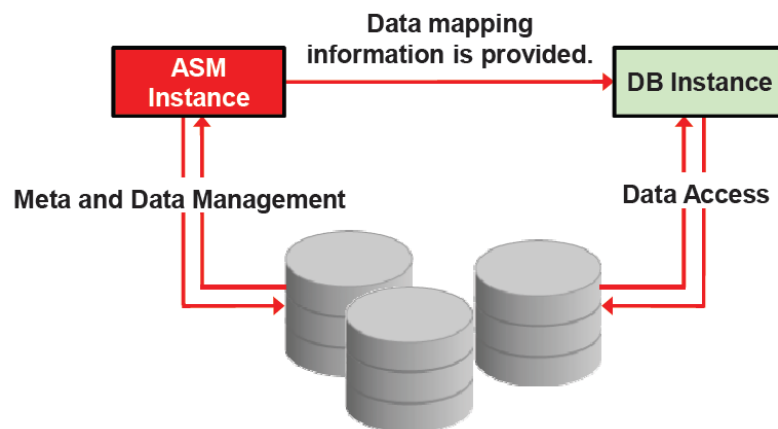
Specifies the transaction name of the Replicat database user so that those transactions are not captured by Extract.

**TRANLOGOPTIONS FILTERTABLE <table\_name>**

Specifies the Replicat checkpoint table name. Operations on the checkpoint table will be ignored by the local Extract.

#### 5.1.4 *ASM and Golden Gate*

Automatic Storage Management (ASM) enables a disk group to be designated for Oracle database files, control files, and backup files.



#### 5.1.4.1 ASM connectivity to Golden Gate



#### 5.1.4.2 Connecting to ASM

TranLogOptions

ASMUser SYS@<ASM\_instance>, ASMPassword <password>

##### **TranLogOptions**

DBLogReader, DBLogReaderBufSize *nnn*

You may use the DBLogReader option with Oracle to cause Extract to use a newer ASM API that is available as of Oracle 10.2.0.5 and later 10g R2 versions, and Oracle 11.2.0.2 and later 11g R2 versions (but not in Oracle 11g R1 versions). This API uses the database server to access the redo and archive logs.

A larger buffer may improve the performance of Extract when redo rate is high. When using DBLogReader, do not use the ASMUser and ASMPassword options of TranLogOptions. The API uses the user and password specified with the UserID parameter.

## 5.2 Configuring Data Pump (Secondary Extract)

### 5.2.1 Data Pump – Overview

- Data is stored in a local trail on the source system
- A second Extract, the data pump:
  - Reads this trail and sends it to one or more targets
  - Manipulates the data or passes it through without change
- Reasons for using
  - A safeguard against network and target failures
  - To break complex data filtering and transformation into phases
  - To consolidate data from many sources
  - To synchronize one source with multiple targets

For most business cases, it is best practice to use a data pump. Some reasons for using a data pump include the following:

- Protection against network and target failures: In a basic GoldenGate configuration, with only a trail on the target system, there is nowhere on the source system to store data that Extract continuously extracts into memory. If the network or the target system becomes unavailable, the primary Extract could run out of memory and abend. However, with a trail and data pump on the source system, captured data can be moved to disk, preventing the abend. When connectivity is restored, the data pump extracts the data from the source trail and sends it to the target system(s).
- You are implementing several phases of data filtering or transformation. When using complex filtering or data transformation configurations, you can configure a data pump to perform the first transformation either on the source system or on the target system, and then use another data pump or the Replicat group to perform the second transformation.
- Consolidating data from many sources to a central target. When synchronizing multiple source databases with a central target database, you can store extracted data on each source system and use data pumps on each system to send the data to a trail on the target system. Dividing the storage load between the source and target systems reduces the need for massive amounts of space on the target system to accommodate data arriving from multiple sources.

- Synchronizing one source with multiple targets. When sending data to multiple target systems, you can configure data pumps on the source system for each one. If network connectivity to any of the targets fails, data can still be sent to the other targets.

### 5.2.2 Data Pump – Configuration

Unified Extract parameter file (containing both primary Extract (capture) and data pump parameters)

```
EXTRACT xtst01
USERID GoldenUser, PASSWORD password
RMTHOST serverx, MGRPORT 7809
RMTRAIL /ggs/dirdat/rt
TABLE SALES.ORDERS;
TABLE SALES.INVENTORY;
```

**With separate data pump process:**

Primary extract (Capture) process parameter file

```
EXTRACT xtst01
USERID GoldenUser, PASSWORD password
EXTTRAIL /ggs/dirdat/aa
TABLE SALES.ORDERS;
TABLE SALES.INVENTORY;
```

Data pump process parameter file

```
EXTRACT ptst01
USERID GoldenUser, PASSWORD password
PASSTHRU
RMTHOST serverx, MGRPORT 7809
RMTRAIL /ggs/dirdat/rt
TABLE SALES.ORDERS;
TABLE SALES.INVENTORY;
```

**PASSTHRU** parameter is used on a data pump if you do not need to do any data transformations or user exit processing.

Add a data pump (source is the local trail from the primary Extract )

```
ADD EXTRACT <datapump>, EXTTRAILSOURCE ./dirdat/<trailid>
```

## 6. Golden Gate Replicat process

### 6.1 Replicat Overview

Replicat can:

- Read data out of GoldenGate trails
- Perform data filtering
- Table, row, operation
- Perform data transformation
- Perform database operations just as your application performed them

GoldenGate trails are temporary queues for the Replicat process. Each record header in the trail provides information about the database change record. Replicat reads these trail files sequentially, and processes inserts, updates and deletes that meet your criteria. Alternatively, you can filter out the rows you do not wish to deliver, as well as perform data transformation prior to applying the data.

Replicat supports a high volume of data replication activity. As a result, network activity is block-based not record-at-a-time. Replicat uses native calls to the database for optimal performance. You can configure multiple Replicat processes for increased throughput.

When replicating, Replicat preserves the boundaries of each transaction so that the target database has the same degree of integrity as the source. Small transactions can be grouped into larger transactions to improve performance. Replicat uses a checkpointing scheme so changes are processed exactly once. After a graceful stop or a failure, processing can be restarted without repetition or loss of continuity.

## 6.2 Configuring Replicat

On the target system:

- Create a checkpoint table in the target database (best practice)
  - **DBLOGIN**
  - **ADD CHECKPOINTTABLE**
- Create a parameter file for Replicat
  - **EDIT PARAMS**
- Add your initial Replicat checkpoint into GoldenGate trails
  - **ADD REPLICAT**
- Start the Replicat process
  - **START REPLICAT**

Replicat reads the GoldenGate trail and applies changes to the target database. Like Extract, Replicat uses checkpoints to store the current read and write position and is added and started using the processing group name.

## 6.3 Replicat – Sample configuration

```
GGSCI> DBLOGIN SOURCEDB mydb USERID login PASSWORD pw
```

```
GGSCI> ADD CHECKPOINTTABLE ggs.checkpt
```

```
GGSCI> EDIT PARAMS REPORD
```

```
REPLICAT REPORD
```

```
USERID ggsuser, PASSWORD ggspass
```

```
ASSUMETARGETDEFS
```

```
DISCARDFILE /ggs/dirrpt/REPORD.dsc, APPEND
```

```
MAP SALES.ORDERS, TARGET SALES.ORDERS;
```



**MAP SALES.INVENTORY, TARGET SALES.INVENTORY;**

**GGSCI> ADD REPLICAT REPO RD, EXTTRAIL /ggs/dirdat/rt**

**GGSCI> START REPLICAT REPO RD**

DBLOGIN USERID and PASSWORD logs the user into the database in order to add the checkpoint table.

**Replicat parameters:**

**TARGETDB** identifies the data source name (not required for Oracle)

**USERID and PASSWORD** provide the credentials to access the database

**ASSUMETARGETS** is used when the source and target systems have the same data definition with identical columns.

**DISCARDFILE** creates a log file to receive records that cannot be processed.

**MAP** establishes the relationship between source table and the target table.

**ADD REPLICAT** names the Replicat group REPO RD and establishes a local trail (EXTTRAIL) with the two-character identifier rt residing on directory dirdat.

## Replicat Parameters

<u>Purpose</u>	<u>Examples</u>
General	SETENV, GETENV, OBEY
Processing method	BEGIN, END, SPECIALRUN
Database login	SOURCEDB, USERID
Selecting, converting and mapping data	COLMATCH, IGNOREUPDATES, MAP, SOURCEDEFS, ASSUMETARGETDEFS
Routing data	EXTFILE, EXTTRAIL
Custom processing	CUSEREXIT, DEFERAPPLYINTERVAL, INCLUDE, MACRO, SQLEXEC
Reporting	REPORT, REPORTCOUNT, STATOPTIONS
Error handling	DISCARDFILE, OVERRIDEDUPS, HANDLECOLLISIONS
Tuning	ALLOCFILES, BATCHSQL, GROUPTRANSOPS, DBOPTIONS
Maintenance	PURGEOLDEXTRACTS, REPORTROLLOVER
Security	DECRYPTTRAIL

### **General**

CHECKPARAMS Performs parameter check, then terminates.

COMMENT Allows insertion of comments in parameter file.

GETENV Retrieves a value for a variable set by SETENV.

OBEY Processes parameter statements contained in a different parameter file.

SETENV Specifies a value for a UNIX environment variable from within the GGSCI interface.

TRACETABLE | NOTRACETABLE Defines a trace table in an Oracle database to which Replicat adds a record whenever it updates the target database.

### **Processing Method**

BEGIN Specifies when Replicat starts processing. Required when SPECIALRUN is used.

BULKLOAD Loads data directly into the interface of Oracle's SQL\*Loader bulkload utility.

END Specifies when Replicat stops processing. Required when using SPECIALRUN.

GENLOADFILES Generates run and control files that are compatible with bulk-load utilities.

REPLICAT Links this run to an online Replicat process.

SPECIALRUN Used for one-time processing that does not require checkpointing from run to run.

## Database Login

TARGETDB Specifies the target database. Support SQL Server, DB2, Sybase and Informix.

USERID Specifies a database user ID and password for connecting to the target database.

## Selecting, Converting and Mapping data

ALLOWDUPTARGETMAP | NOALLOWDUPTARGETMAP Allows the same source-target MAP statement to appear more than once in the parameter file.

ASCIITOEBCDIC Converts incoming ASCII text to EBCDIC for DB2 on z/OS systems running UNIX System Services.

ASSUMETARGETDEFS Assumes the source and target tables have the same column structure.

COLMATCH Establishes global column-mapping rules.

DDL Enables and filters the capture of DDL operations.

DDLSUBST Enables string substitution in DDL processing.

FILTERDUPS | NOFILTERDUPS Controls the handling of anomalies in data that is sent out of order from NSK.

GETDELETES | IGNOREDELETES Includes (default) or excludes delete operations from being replicated.

GETINSERTS | IGNOREINSERTS Controls the replication of insert operations.

GETUPDATEAFTERS | IGNOREUPDATEAFTERS Controls the replication of update after images.

GETUPDATEBEFORES | IGNOREUPDATEBEFORES Controls the replication of update before images.

GETUPDATES | IGNOREUPDATES Controls the replication of update operations.

GETTRUNCATES | IGNORETRUNCATES Includes or excludes the processing TRUNCATE TABLE operations. Default is IGNORETRUNCATES.

INSERTALLRECORDS Inserts all records, including before and after images, as inserts into the target databases.

INSERTDELETES | NOINSERTDELETES Converts deletes to inserts.

INSERTMISSINGUPDATES | NOINSERTMISSINGUPDATES Converts an updates to an insert when a target row does not exist.

INSERTUPDATES | NOINSERTUPDATES Converts updates to inserts.

MAP Specifies a relationship between one or more source and target tables and controls column mapping and conversion.

MAPEXCLUDE Excludes tables from being processed by a wildcard specification supplied in MAP statements.

REPLACEBADCHAR Replaces invalid character values with another value.

REPLACEBADNUM Replaces invalid numeric values with another value.

SOURCEDEFS Specifies a file that contains source table data definitions created by the DEFGEN utility.

SPACESTONULL | NOSPACESTONULL Controls whether or not a target column containing only spaces is converted to NULL on Oracle.

TABLE (Replicat) Specifies a table or tables for which event actions are to take place when a row satisfies the given filter criteria.

TRIMSPACES | NOTRIMSPACES Controls if trailing spaces are preserved or removed for character or variable character columns.

UPDATEDELETES | NOUPDATEDELETES Changes deletes to updates.

USEDATEPREFIX Prefixes data values for DATE data types with a DATE literal, as required by Teradata databases.

USETIMEPREFIX Prefixes data values for TIME datatypes with a TIME literal, as required by Teradata databases.

USETIMESTAMPPREFIX Prefixes data values for TIMESTAMP datatypes with a TIMESTAMP literal, as required by Teradata databases.

### **Routing Data**

EXTFILE Defines the name of an extract file on the local system that contains data to be replicated. Used for one-time processing.

EXTTRAIL Defines a trail containing data to be replicated Used for one-time processing.

### **Custom Processing**

CUSEREXIT Invokes customized user exit routines at specified points during processing.

DEFERAPPLYINTERVAL Specifies a length of time for Replicat to wait before applying replicated operations to the target database.

INCLUDE References a macro library in a parameter file.

MACRO Defines a GoldenGate macro.

SQLEXEC Executes a stored procedure, query or database command during Replicat processing.

### **Reporting**

CMDTRACE Displays macro expansion steps in the report file.

LIST | NOLIST Suppresses or displays the listings of the parameter file to the report file.

REPORT Schedules a statistical report at a specified date or time.

REPORTCOUNT Reports the number or records processed.

SHOWSYNTAX Causes Replicat to print its SQL statements to the report file.

STATOPTIONS Specifies information to include in statistical displays.

TRACE | TRACE2 Shows Replicat processing information to assist in revealing processing bottlenecks.

### **Error Handling**

CHECKSEQUENCEVALUE | NOCHECKSEQUENCEVALUE (Oracle) Controls whether or not Replicat verifies that a target sequence value is higher than the one on the source and corrects any disparity that it finds.

DDLERROR Controls error handling for DDL replication.

DISCARDFILE Contains records that could not be processed.

HANDLECOLLISIONS | NOHANDLECOLLISIONS Handles errors for duplicate and missing records. Reconciles the results of changes made to the target database by an initial load process with those applied by a change-synchronization group.

HANDLETPKUPDATE Prevents constraint errors associated with replicating transient primary key updates.

OVERRIDEDUPS | NOOVERRIDEDUPS Overlays a replicated insert record onto an existing target record whenever a duplicate-record error occurs.

RESTARTCOLLISIONS | NORESTARTCOLLISIONS Controls whether or not Replicat applies HANDLECOLLISIONS logic after GoldenGate has exited because of a conflict.

REPERORR Determines how Replicat responds to database errors.

REFETCHEDCOLOPTIONS Determines how Replicat responds to operations for which a fetch from the source database was required.

SQLDUPERR When OVERRIDEDUPS is on, specifies the database error number that indicates a duplicate-record error.

WARNRATE Determines how often database errors are reported.

## **Tuning**

ALLOCFILES Used to control the incremental amount of memory structured allocated once the initial allocation specified by NUMFILES parameter is reached. Defaults to 500.

BATCHSQL Increases the throughput of Replicat processing by arranging similar SQL statements into arrays and applying them at an accelerated rate.

CHECKPOINTSECS Controls how often Replicat writes a checkpoint when checkpoints are not being generated as a result of transaction commits.

DBOPTIONS Specifies database options.

DDLOPTIONS Specifies DDL processing options.

DYNAMICRESOLUTION | NODYNAMICRESOLUTION Makes Replicate start faster when there is a large number of tables specified for synchronization.

DYNSQL | NODYNSQL Causes Replicat to use literal SQL statements rather than a compile-once, execute-many strategy.

EOFDELAY | EOFDELAYSECS Determines how many seconds Replicat delays before looking for more data to process.

FUNCTIONSTACKSIZE Controls the number of arguments permitted when using GoldenGate column-conversion functions.

GROUPTRANSOPS Groups multiple transactions into larger transactions.

INSERTAPPEND | NOINSERTAPPEND Controls whether or not Replicat uses an APPEND hint when applying INSERT operations to Oracle target tables.

LOBMEMORY Controls the memory and disk space allocated to LOB transactions.

MAXDISCARDRECS Limits the number of discarded records reported to the discard file.

MAXSQLSTATEMENTS Controls the number of prepared SQL statements that can be used by Replicat.

MAXTRANSOPS Divides large transactions into smaller ones.

NUMFILES Used to control the initial amount of memory structured allocate for the storage of tables structures. Defaults to 1000.

RETRYDELAY Specifies the delay between attempts to retry a failed SQL operation.

TRANSACTIONTIMEOUT Specifies a time interval after which Replicat will commit its open target transaction and roll back any incomplete source transactions that it contains, saving them for when the entire source transaction is ready to be applied.

TRANSMEMORY (DB2 z/OS and NonStop SQL/MX only) Controls the amount of memory and temporary disk space available for caching uncommitted transaction data.

WILDCARDRESOLVE Alters the rules by which wildcard specifications in a MAP statement are resolved.

### **Maintenance**

DISCARDROLLOVER Specifies when to create new discard files.

PURGEOLDEXTRACTS Purges GoldenGate trail files once consumed.

REPORTROLLOVER Specifies when to create new report files.

### **Security**

DECRYPTTRAIL Decrypts data in a trail or extract file.

## 7. Trail files and Report files

### 7.1 Overview of Extract Trails and Files

- Extract writes data to any of the following:
  - Local trail (ExtTrail) on the local system
  - Local file (ExtFile) on the local system
  - Remote trail (RmtTrail) on a remote system
  - Remote file (RmtFile) on a remote system
- Extract trails and files are unstructured, with variable length records.
  - I/O is performed using large block writes.
- Extract writes checkpoints for trails during change capture:
  - This guarantees that no data is lost during restart.
  - Multiple Replicat processes may process the same trail.

### 7.2 Extract Trails and Files Contents

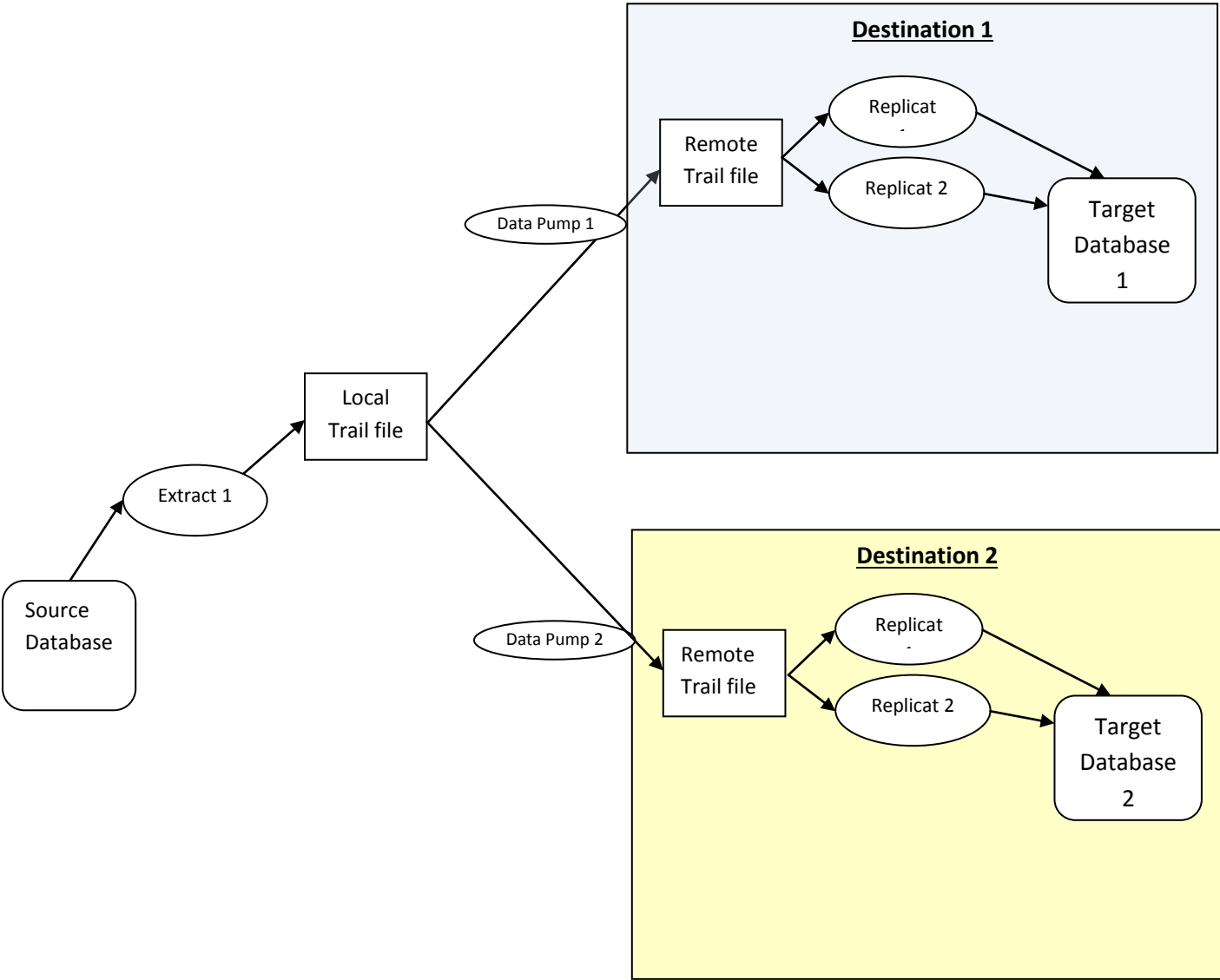
- Each record in the trail contains an operation that has been committed in the source database.
- Committed transactional order is preserved.
- Operations in a transaction are grouped together in the order in which they were applied.
- By default, only the primary key and changed columns are recorded.
- Flags indicate the first and last records in each transaction.

### 7.3 Extract Trails and Files Cleanup

Trail files can be purged after they are consumed:

- The temporary storage requirement is small if processes keep pace.
- Configure Manager to purge used trail data (best practice).

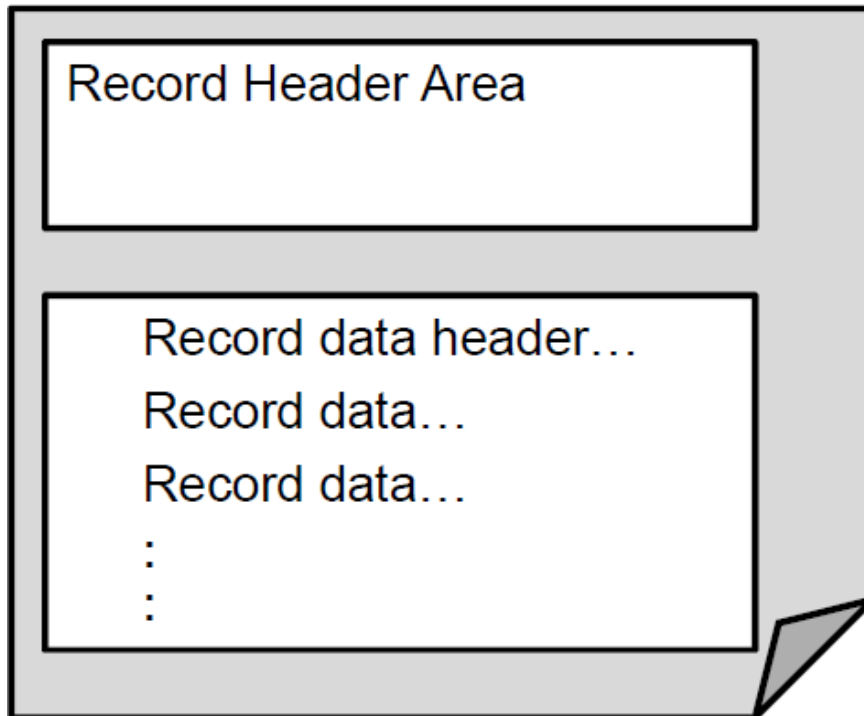
# 7.4 Parallelism in Processes





## 7.5 GoldenGate Data Format - File Header

Each Golden Gate trail file contains:



Each trail file has a file header that contains:

Trail file information

- Compatibility level
- Character set
- Creation time
- File sequence number
- File size

First and last record information

- Timestamp
- Commit sequence number (CSN)

Extract information

- GoldenGate version
- Group name
- Host name
- Hardware type
- OS type and version
- DB type, version and character set

## 7.6 GoldenGate Data Format - Compatibility Level

- Identifies the trail file format by GoldenGate <major>.<minor> version numbers
- Allows customers to use different versions of GoldenGate Extract, trail files and Replicat together
- Set in Extract EXTFILE, EXTTRAIL, RMTFILE or RMTTRAIL parameter; for example:
  - **RMTTRAIL /ggs/dirdat/ex, FORMAT RELEASE 10.0**
- The input and output trails of a data pump must have the same compatibility level

## 7.7 GoldenGate Data Format – Records

Each trail record contains:

- Eye-catchers for each section
- GoldenGate record header (50 bytes plus length of table name)
  - Contains metadata of the change
    - Table or File name
    - I/O type
    - Before/After indicator
    - Transaction information
    - Transaction time
    - Transaction group
    - Length of data area
    - Plus other information
- Optional user token area
  - Token ID, token value
- Data area
  - Column ID, column value

## Record Header

Use the Logdump utility to examine the record header. Here is a layout of the header record:

- **Hdr-Ind:** Always E, indicating that the record was created by the Extract process.
- **UndoFlag:** (NonStop) Normally, UndoFlag is set to zero, but if the record is the backout of a previously successful operation, then UndoFlag will be set to 1.
- **RecLength:** The length, in bytes, of the record buffer.
- **IOType:** The type of operation represented by the record.
- **TransInd:** The place of the record within the current transaction. Values are 0 for the first record in transaction; 1 for neither first nor last record in transaction; 2 for the last record in the transaction; and 3 for the only record in the transaction.
- **SyskeyLen:** (NonStop) The length of the system key (4 or 8 bytes) if the source is a NonStop file and has a system key.
- **AuditRBA:** The relative byte address of the commit record.
- **Continued:** (Windows and UNIX) Identifies (Y/N) whether or not the record is a segment of a larger piece of data that is too large to fit within one record, such as LOBs.
- **Partition:** Depends on the record type and is used for NonStop records. In the case of BulkIO operations, Partition indicates the number of the source partition on which the bulk operation was performed. For other non-bulk NonStop operations, the value can be either 0 or 4. A value of 4 indicates that the data is in FieldComp format.
- **BeforeAfter:** Identifies whether the record is a before (B) or after (A) image of an update operation. **OrigNode:** (NonStop) The node number of the system where the data was extracted.
- **FormatType:** Identifies whether the data was read from the transaction log (R) or fetched from the database (F).
- **AuditPos:** Identifies the position of the Extract process in the transaction log.
- **RecCount:** (Windows and UNIX) Used to reassemble LOB data when it must be split into 2K chunks to be written to the GoldenGate file.

## 7.8 Extract Trails and Files - Alternative Formats

Alternative output formats can be specified by the Extract parameters:

- FORMATASCII
- FORMATSQ
- FORMATXML

### 7.8.1 Alternative Formats: *FORMATASCII*

- Output is in external ASCII format
- Can format data for popular database load utilities
  - FORMATASCII, BCP
  - FORMATASCII, SQLLOADER
- Data cannot be processed by GoldenGate Replicat

**Example 1. FORMATASCII without options produces the following:**

```
B,1997-02-17:14:09:46.421335,8,1873474,I,A, TEST.CUSTOMER
,CUSTNAME,'Eric',LOCATION,'San Fran',BALANCE,550,V,A,
TEST.CUSTOMER,CUSTNAME,'Eric',BALANCE,100,C,
```

**Example 2. FORMATASCII, NONAMES, DELIMITER '|' produces the following:**

```
B|1997-02-17:14:09:46.421335|8|1873474|
I|A|CUSTOMER|'Eric'|'San Fran'|550|
V|A|CUSTOMER|CUSTNAME|'Eric'|BALANCE|100|C|
```

### 7.8.2 Alternative Formats: *FORMATSQL*

- Output is in external SQL DML format
- Data cannot be processed by Replicat
- Default output for each transaction includes:
  - Begin transaction indicator, B
  - Timestamp at which the transaction was committed
  - Sequence number of transaction log containing commit
  - RBA of commit record within transaction log

- SQL Statement
- Commit indicator, C
- Newline indicator

**Example 1: FORMATSQ**

```
B,2008-11-11:13:48:49.000000,1226440129,155,
DELETE FROM TEST.TCUSTMER WHERE CUST_CODE='JANE';
DELETE FROM TEST.TCUSTMER WHERE CUST_CODE='WILL';
DELETE FROM TEST.TCUSTORD WHERE CUST_CODE='JANE' AND
ORDER_DATE='1995-11-11:13:52:00' AND PRODUCT_CODE='PLANE' AND
ORDER_ID='256';
DELETE FROM TEST.TCUSTORD WHERE CUST_CODE='WILL' AND
ORDER_DATE='1994-09-30:15:33:00' AND PRODUCT_CODE='CAR' AND
ORDER_ID='144';
INSERT INTO TEST.TCUSTMER (CUST_CODE,NAME,CITY,STATE) VALUES
('WILL','BG SOFTWARE CO.','SEATTLE','WA');
INSERT INTO TEST.TCUSTMER (CUST_CODE,NAME,CITY,STATE) VALUES
('JANE','ROCKY FLYER INC.','DENVER','CO');
INSERT INTO TEST.TCUSTORD
(CUST_CODE,ORDER_DATE,PRODUCT_CODE,ORDER_ID,PRODUCT_PRICE,PRODU
C
T_AMOUNT,TRANSACTION_ID) VALUES ('WILL','1994-09-
30:15:33:00','CAR','144',17520.00,3,'100');
INSERT INTO TEST.TCUSTORD
(CUST_CODE,ORDER_DATE,PRODUCT_CODE,ORDER_ID,PRODUCT_PRICE,PRODU
C
T_AMOUNT,TRANSACTION_ID) VALUES ('JANE','1995-11-
11:13:52:00','PLANE','256',133300.00,1,'100');
C,
```

### 7.8.3 *Alternative Formats: FORMATXML*

- Output is in XML format
- Data cannot be processed by Replicat

#### **Example 1: FORMATXML**

```
<transaction timestamp="2008-11-11:14:33:12.000000">
<dbupdate table="TEST.TCUSTMER" type="insert">
<columns>
<column name="CUST_CODE" key="true">ZEKE</column>
<column name="NAME">ZEKE'S MOTION INC.</column>
<column name="CITY">ABERDEEN</column>
<column name="STATE">WA</column>
</columns>
</dbupdate>
<dbupdate table="TEST.TCUSTMER" type="insert">
<columns>
<column name="CUST_CODE" key="true">ZOE</column>
<column name="NAME">ZOE'S USED BICYCLES</column>
<column name="CITY">ABERDEEN</column>
<column name="STATE">WA</column>
</columns>
</dbupdate>
<dbupdate table="TEST.TCUSTMER" type="insert">
<columns>
<column name="CUST_CODE" key="true">VAN</column>
<column name="NAME">VAN'S BICYCLESS</column>
<column name="CITY">ABERDEEN</column>
<column name="STATE">WA</column>
</columns>
</dbupdate>
</transaction>
```

## 7.9 Logdump

The Logdump utility allows you to:

- Display or search for information that is stored in GoldenGate trails or files
- Save a portion of a GoldenGate trail to a separate trail file

To start Logdump - from the GoldenGate installation directory:

```
Shell> logdump
```

To get help:

```
Logdump 1 > help
```

```
Logdump> open dirdat/rt000000
```

Logdump responds with:

```
Current LogTrail is /ggs/dirdat/rt000000
```

To view the trail file header:

```
Logdump 1> fileheader on
```

To view the record header with the data:

```
Logdump 2> ghdr on
```

To add column information:

```
Logdump 3> detail on
```

To go to the first record, and to move from one record to another in sequence:

```
Logdump 6 > pos 0
```

```
Logdump 7 > next (or just type n)
```

To position at an approximate starting point and locate the next good header record:

```
Logdump 8 > pos <approximate RBA>
```

```
Logdump 9 > scanforheader (or just type sfh)
```

## 7.9.1 Logdump – Viewing Trail Records

Record header: contains transaction information.  
Below the header is the data area.

I/O type

Operation type and time the record was written

Source table

Image type: could be a before or after image

Column information

Record data, in hex

Length of record and its RBA position in the trail file

Record data, in ASCII

```

Hdr-Ind      : E (x45)      Partition   :      (x04)
UndoFlag     :      (x00)    BeforeAfter : 0 (x41)
RecLength    : 28 (x001c)   IO Time     : 2005/10/27 14:33:57.000.000
IOType       : 5 (x05)      OrigNode    : 255 (xff)
TransInd     : 0 (x00)      FormatType  : R (x52)
SyskeyLen    : 0 (x00)      Incomplete  :      (x00)
AuditRBA     :      16      AuditPos    : 42923848
Continued    : N (x00)      RecCount    : 1 (x01)

2005/10/27 14:33:57.000.000 Insert                               Len 28 RBA 0
Name: HR.REGIONS
After Image:
0000 0000 0000 3FFE 0000 0000 0000 0001 0000 0000 | .....?.....
0006 4575 726F 7065 | ..Europe
Column 0 (x0000) Len 10 (x000a)
Column 1 (x0001) Len 10 (x000a)

```

GoldenGate trail files are unstructured. The GoldenGate record header provides metadata of the data contained in the record and includes the following information.

- The operation type, such as an insert, update, or delete
- The transaction indicator (TransInd): 00 beginning, 01 middle, 02 end or 03 whole of transaction
- The before or after indicator for updates
- Transaction information, such as the transaction group and commit timestamp
- The time that the change was written to the GoldenGate file
- The type of database operation
- The length of the record
- The relative byte address within the GoldenGate file
- The table name

The change data is shown in hex and ASCII format. If before images are configured to be captured, for example to enable a procedure to compare before values in the WHERE clause, then a before image also would appear in the record.



## 7.9.2 Logdump – Filtering on a Filename

```
Logdump 7 >filter include filename TCUST*
```

```
Logdump 8 >filter match all
```

```
Logdump 9 >n
```

---

```
Hdr-Ind : E (x45) Partition : . (x00)
UndoFlag : . (x00) BeforeAfter: A (x41)
RecLength : 56 (x0038) IO Time : 2002/04/30 15:56:40.814
IOType : 5 (x05) OrigNode : 108 (x6c)
TransInd : . (x01) FormatType : F (x46)
SyskeyLen : 0 (x00) Incomplete : . (x00)
AuditRBA : 105974056
2002/04/30 15:56:40.814 Insert Len 56 Log RBA 1230
File: TCUSTMER Partition 0
After Image:
3220 2020 4A61 6D65 7320 2020 2020 4A6F 686E 736F | 2 James Johnso
6E20 2020 2020 2020 2020 2020 2020 4368 6F75 6472 | n Choudr
616E 7420 2020 2020 2020 2020 2020 4C41 | LA
Filtering suppressed 18 records
```

Use FILTER to filter the display based on one or more criteria.

- FILENAME specifies a SQL table, NonStop data file or group of tables/files using a wildcard.
- You can string multiple FILTER commands together, separating each one with a semi-colon, as in:

```
FILTER INCLUDE FILENAME fin.act*; FILTER RECTYPE 5; FILTER MATCH ALL
```

- To avoid unexpected results, avoid stringing filter options together with one FILTER command. For example, the following would be *incorrect*:

```
FILTER INCLUDE FILENAME fin.act*; RECTYPE 5; MATCH ALL
```

- Without arguments, FILTER displays the current filter status (ON or OFF) and any filter criteria that are in effect.

The FILTER command can INCLUDE | EXCLUDE the following options:

```
AUDITRBA <rba> [<comparison operator>]
CLEAR {<filter_spec> | ALL}
ENDTIME <time_string>
FILENAME <name> [, <name>]
HEX <hex_string> [<byte_range>] [, ...]
INT16 <16-bit_integer> | INT32 <32-bit_integer>
IOTYPE <operation type> [, <operation type>]
MATCH {ANY | ALL}
```

OFF  
ON  
PROCESS <process\_name>  
RBA <byte address> [<comparison operator>] [...]  
RECLEN <length> [<comparison operator>]  
RECTYPE {<type\_number> | <type\_name>}  
SHOW  
STARTTIME <time\_string>  
STRING [BOTH] [B],<text> [<column\_range>][[B],<text> [<column\_range>]] [...]  
SYSKEY <system key> [<comparison operator>] [...]  
TRANSIND <indicator> [<comparison operator>]  
TYPE <type>  
UNDOFLAG <type> [<comparison operator>]

### **7.9.3 Logdump – Saving Records to a New Trail**

*Logdump> save newtrail 10 records*

The 10 records are taken forward from the current position in the file.

### **7.9.4 Logdump – Keeping a Log of Your Session**

*Logdump> log to MySession.txt*

*When finished...*

*Logdump> log stop*

## 7.10 Reversing the Trail Sequence

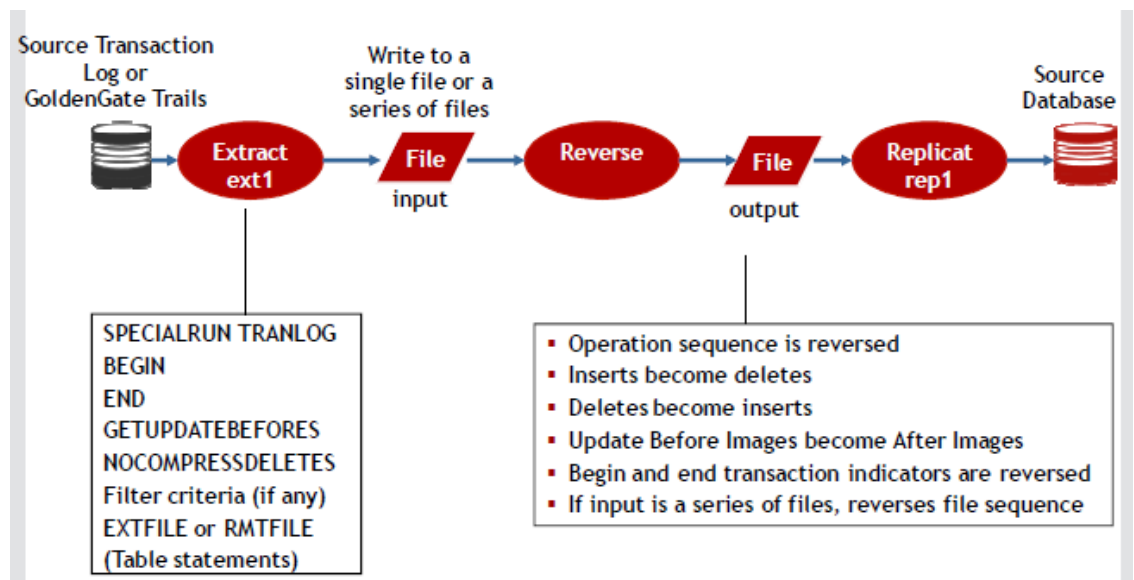
### 7.10.1 Reverse – Overview

The Reverse utility reorders operations within GoldenGate trails in reverse sequence:

- Provides selective back out of operations
  - Selectively back out corrupt data or accidental delete operations while keeping the rest of the application alive
- Can be used to restore a database to a specific point in time
  - Use to back out all operations during regression testing to restore the original test baseline

To use GoldenGate Reverse:

- Run Extract to extract the before data.
- Run the Reverse utility to perform the reversal of the transactions.
- Run Replicat to apply the restored before data to the target database.



### 7.10.1.1 Reverse - Example

#### Extract Parameters (dirprm/ext1.prm)

```
SPECIALRUN, TRANLOG
SOURCDB dsn, USERID user, PASSWORD password
BEGIN 2004-05-30 17:00
END 2004-05-30 18:00
GETUPDATEBEFORES
NOCOMPRESSDELETES
RMTHOST target, MGRPORT 7809
RMTFILE /ggs/dirdat/input.dat
TABLE HR.SALES;
TABLE HR.ACCOUNTS;
```

#### Replicat Parameters (dirprm/rep1.prm)

```
SPECIALRUN
END RUNTIME
TARGETDB dsn, USERID user, PASSWORD password
EXTFILE /ggs/dirdat/output.dat
ASSUMETARGETDEFS
MAP HR.SALES, TARGET HR.SALES;
MAP HR.ACCOUNTS, TARGET HR.ACCOUNTS;
```

1. Run Extract from either GoldenGate trails or the source database transaction logs  
**\$ ggs/> extract paramfile dirprm/ext1.prm**
2. Run Reverse to produce the reordered output  
**\$ ggs/> reverse dirdat/input.dat dirdat/output.dat**
3. Run Replicat to back out operations  
**\$ ggs/> replicat paramfile dirprm/rep1.prm**

## 8. Initial Load

An initial load takes a copy of the entire source data set, transforms it if necessary, and applies it to the target tables so that the movement of transaction data begins from a synchronized state.

The first time that you start change synchronization will be during the initial load process. Change synchronization keeps track of ongoing transactional changes while the load is being applied.

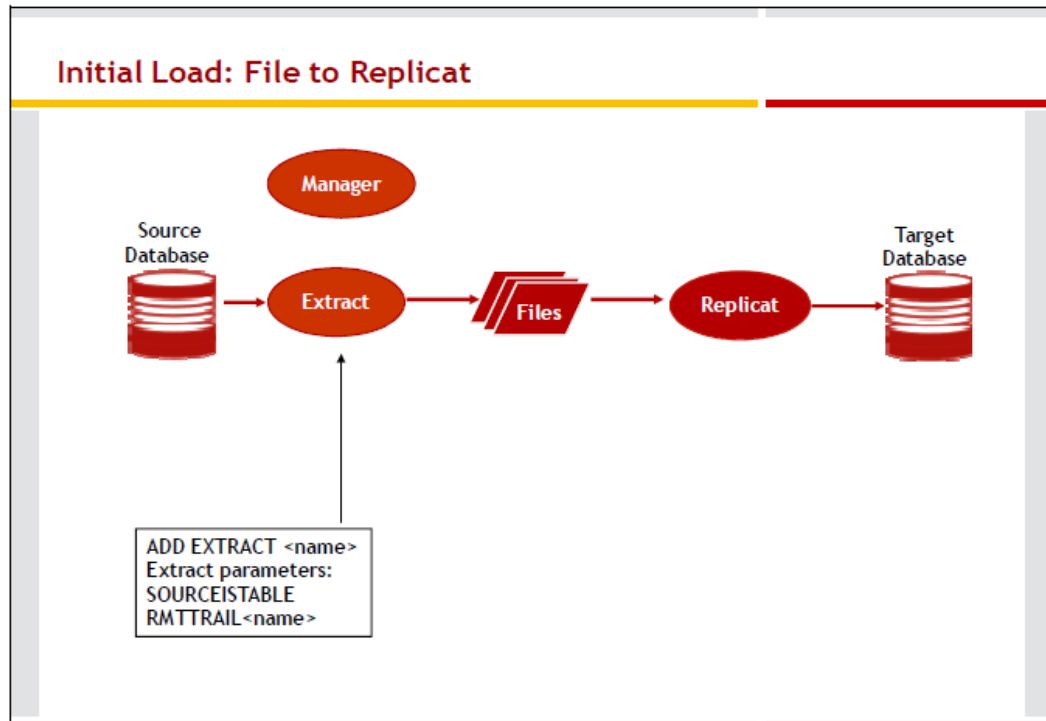
### 8.1 Golden Gate Initial Load Methods

GoldenGate method	Extract writes to	Load method
File to Replicat	Trail (GoldenGate format)	Replicat via SQL
File to database utility	Formatted text file	Database utility
Direct load	Replicat (directly)	Replicat via SQL
Direct bulk load	Replicat (directly)	Replicat via SQL*Loader API

Initial Load is similar to database specific methods like:

- Backup/Restore
- Export/Import
- SQL scripts
- Break mirror
- Transportable tablespaces (Oracle)

## 8.2 Initial Load methodology



### File to Replicat

Captures data directly from the source tables and writes to an Extract file for Replicat to process.

### Extract parameters

`SOURCEISTABLE` instructs Extract to read the source tables directly rather than the transaction log.

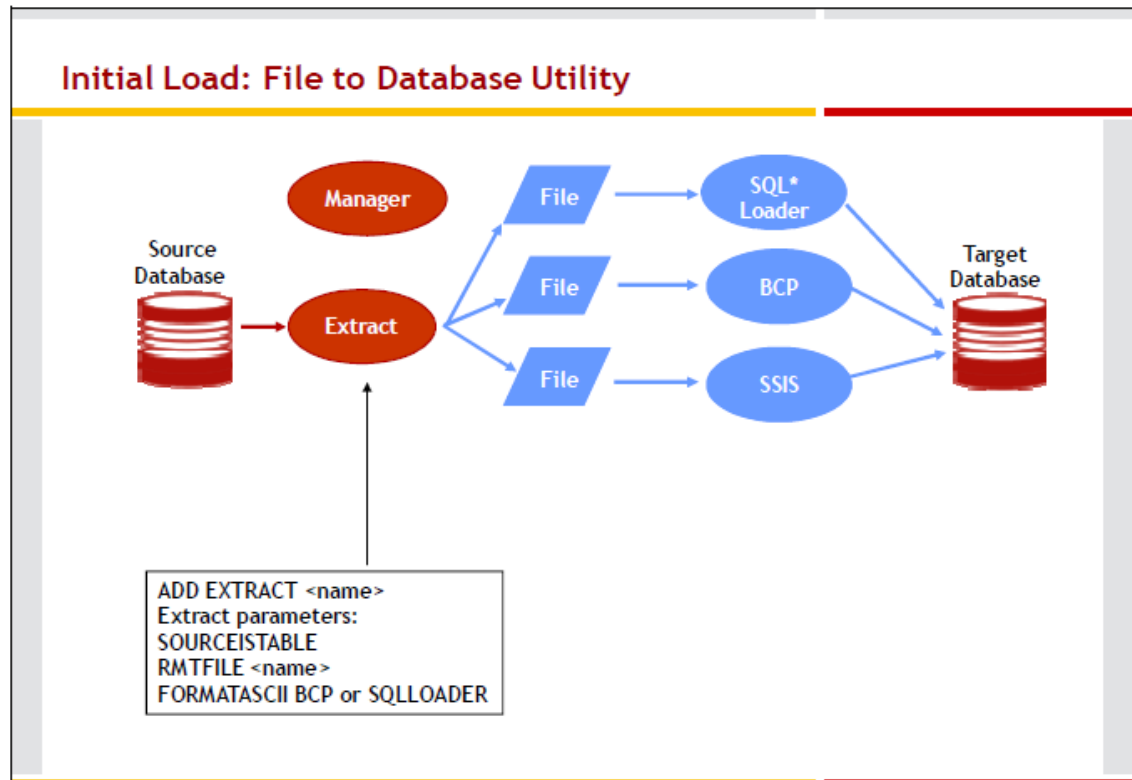
To format the output for processing by Replicat, use `RMTTRAIL`. Using Replicat provides the ability to perform additional data transformation prior to loading the data.

### Execution

You can start Extract by the GGSCI command: `START EXTRACT <name>`

Or from the command shell with syntax:

```
extract paramfile <command file> [ reportfile <out file> ]
```



#### File to Database Utility

Captures data directly from the source tables and outputs the data to files for processing by native bulk loaders.

#### Extract parameters

SOURCEISTABLE instructs Extract to read the source tables directly rather than the transaction log.

To format the output for native bulk utilities, such as SSIS, BCP, or SQL\*Loader, use:

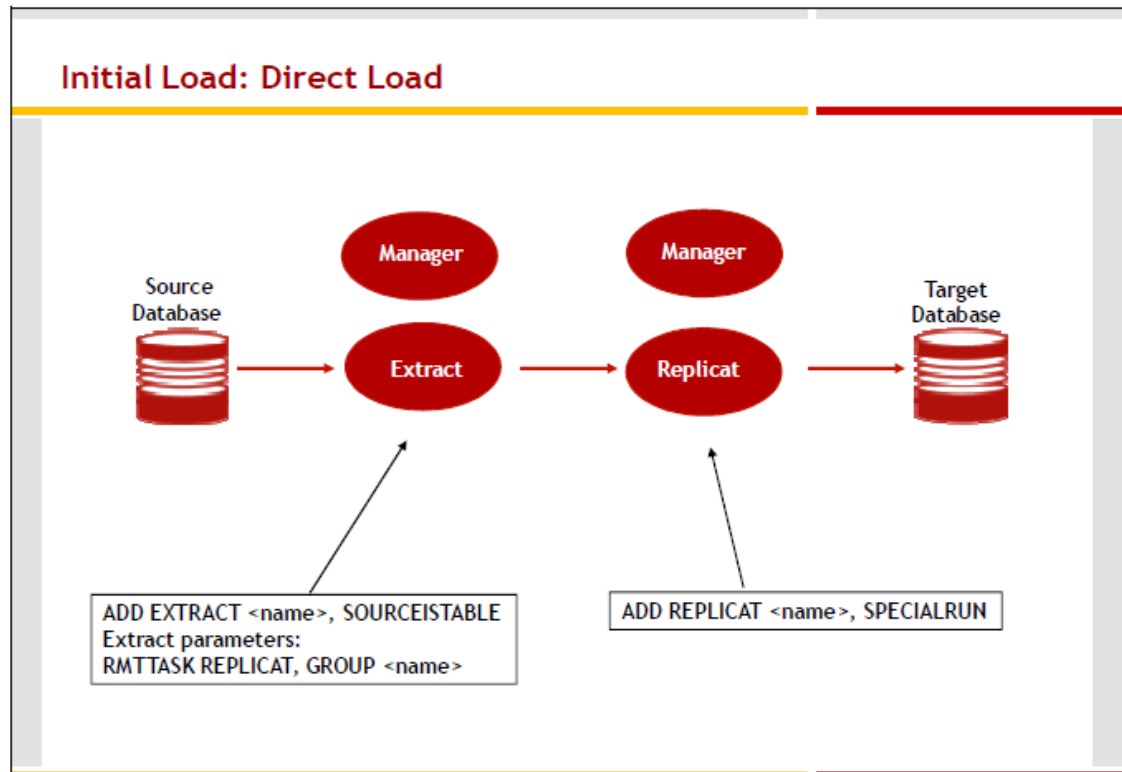
RMTFILE FORMATASCII with appropriate options, like BCP or SQLLOADER

#### Execution

You can start Extract by the GGSCI command: START EXTRACT <name>

Or from the command shell with syntax:

```
extract paramfile <command file> [ reportfile <out file> ]
```



### Direct Load

Captures data directly from the source tables and sends the data in large blocks to the Replicat process.

Using Replicat provides the ability to perform additional data transformation prior to loading the data.

### Extract parameters

Here you have RMTTASK (instead of RMTFILE in the Queue Data method).

RMTTASK instructs the Manager process on the target system to start a Replicat process with a group name specified in the GROUP clause.

### Execution

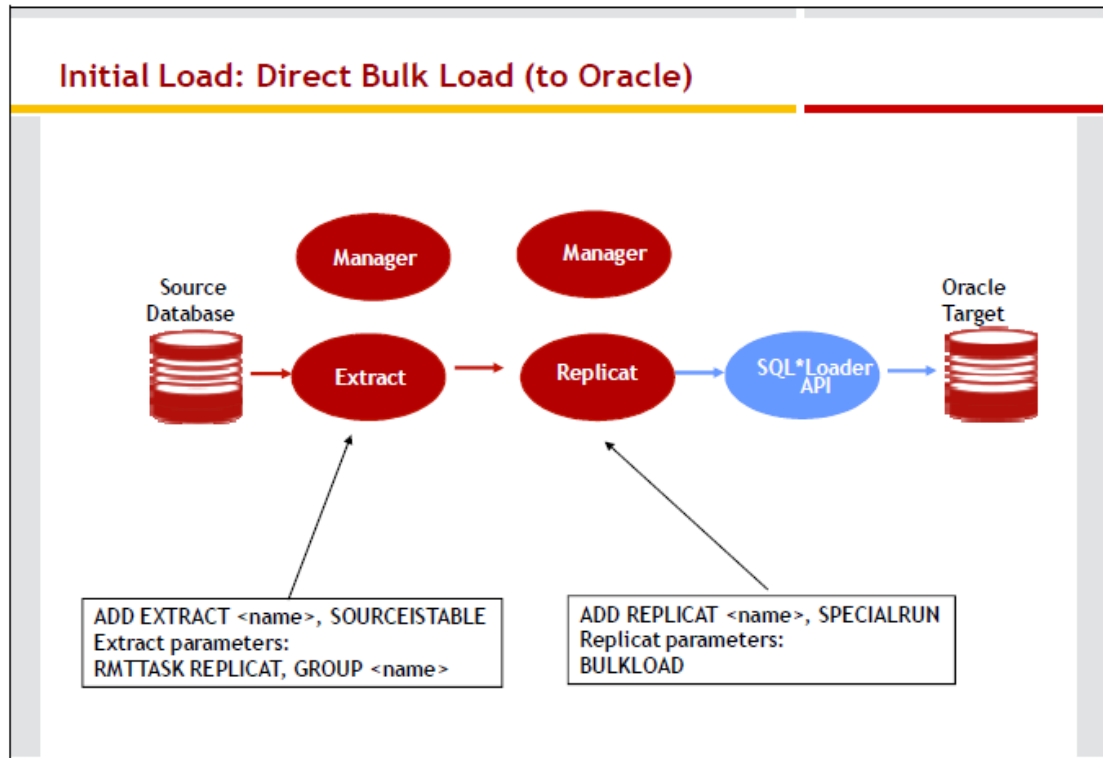
When you add Extract and Replicat:

- SOURCEISTABLE instructs Extract to read the source tables directly rather than the transaction log.
- SPECIALRUN on Replicat specifies one-time batch processing where checkpoints are not maintained.

The initial data load is then started using the GGSCI command START EXTRACT.



The Replicat process will be automatically started by the Manager process. The port used by the Replicat process may be controlled using the DYNAMICPORTLIST Manager parameter.



### Direct Bulk Load

The Direct Bulk Load method is the fastest method using GoldenGate for initial data load. It sends data in large blocks to the Replicat process, which communicates directly with SQL\*Loaders through an API. The Manager process dynamically starts the Replicat process. Using Replicat provides the ability to perform additional data transformation prior to loading the data. The port used by the Delivery process may be controlled using the DYNAMICPORTLIST Manager parameter.

### Extract parameters

Here you have RMTTASK (instead of RMTFILE in the Queue Data method).

RMTTASK instructs the Manager process on the target system to start a Replicat process with a group name specified in the GROUP clause.

### Replicat parameters

The parameter BULKLOAD distinguishes this from the direct load method.

## Execution

When you add Extract and Replicat:

- SOURCEISTABLE instructs Extract to read the source tables directly rather than the transaction log.
- SPECIALRUN on Replicat specifies one-time batch processing where checkpoints are not maintained.

The initial data load is then started using the GGSCI command START EXTRACT.

The Replicat process will be automatically started by the Manager process.

The port used by the Replicat process may be controlled using the DYNAMICPORTLIST Manager parameter.

## 8.3 Collision handling when using initial load

### 8.3.1 *Avoiding Collisions with Initial Load*

1. After the initial load completes, note the highest CSN number of the standby database.
2. Start Replicat to read from the next CSN:

Start Replicat *<group>* AtCSN *<csn>* | AfterCSN *<csn>* | SkipTransaction

- **AtCSN *<csn>*** causes Replicat to skip transactions in the trail until it finds a transaction that contains the specified CSN. *<csn>* must be in the format that is native to the database.

- **AfterCSN *<csn>*** causes Replicat to skip transactions in the trail until it finds the first transaction after the one that contains the specified CSN.

- **SkipTransaction** causes Replicat to skip the first transaction in the trail after startup. All operations in that first transaction are excluded.

### 8.3.2 Handling Collisions with Initial Load

If you cannot avoid collisions, you must *handle* them.

- The Replicat **HandleCollisions** parameter can be used.
  - When Replicat encounters a duplicate-record error on an insert, it writes the change record over the initial data load record.
  - When Replicat encounters a missing-record error for an update or delete, the change record is discarded.
- After all of the change data generated during the load has been replicated, turn off **HandleCollisions**.

GGSCI> **Send Replicat** *<group>* **NoHandleCollisions** (for running process)

GGSCI> **Edit Param** *<group>* (To permanently remove HandleCollisions parameter)

## 9. Data selection, filtering and transformation

### 9.1 Overview

- Data selection and filtering
- Column mapping
- Functions
- SQLEXEC
- Macros
- User tokens
- User exits
- Oracle Sequences

### 9.2 Data selection and filtering

GoldenGate provides the ability to select or filter out data based on a variety of levels and conditions.

Parameter / Clause	Selects
TABLE or MAP	Table
WHERE	Row
FILTER	Row, Operation, Range
TABLE COLS   COLSEXCEPT	Columns

- **TABLE selection**

The MAP (Replicat) or TABLE (Extract) parameter can be used to select a table.

***MAP sales.tcustord, TARGET sales.tord;***

- **ROWS selection**

The following WHERE option can be used with MAP or TABLE to select rows for “AUTO” product type.

***WHERE (PRODUCT\_TYPE = “AUTO”);***

- **OPERATIONS selection**

The following can be used with MAP or TABLE to select rows with amounts greater than zero only for update and delete operations.

***FILTER (ON UPDATE, ON DELETE, amount > 0);***

- **COLUMNS selection**

The COLS and COLSEXCEPT options of the TABLE parameter allow selection of columns as shown in the example below. Use COLS to select columns for extraction, and use COLSEXCEPT to select all columns except those designated by COLSEXCEPT, for example:

***TABLE sales.tcustord, TARGET sales.tord, COLSEXCEPT (facility\_number);***

### **9.2.1 Data Selection – WHERE Clause**

WHERE clause appears on either the MAP or TABLE parameter and must be surrounded by parenthesis

WHERE clause cannot:

- perform arithmetic operations
- refer to trail header and user token values

**WHERE** can perform an evaluation for:

<b>Element Description</b>	<b>Example</b>
Columns	PRODUCT_AMT
Comparison operators	=, <>, >, <, >=, <=
Numeric values	-123, 5500.123
Literal strings	"AUTO", "Ca"
Field tests	@NULL,@PRESENT,@ABSENT
Conjunctive operators	AND, OR

Only rows where the state column has a value of CA are returned.

***WHERE (STATE = "CA");***

Only rows where the amount column has a value of NULL. Note that if amount was not part of the update, the result is false.

***WHERE (AMOUNT = @NULL);***

Only rows where the amount was part of the operation and it has value that is not null.

***WHERE (AMOUNT @PRESENT AND AMOUNT <> @NULL);***

Only rows where the account identifier is greater than CORP-ABC.

***WHERE (ACCOUNT\_ID > "CORP-ABC");***

## 9.2.2 Selection – FILTER Clause

### 9.2.2.1 FILTER Overview

- FILTER clause appears on either the MAP or TABLE parameter and must be surrounded by parenthesis
- With FILTER you can:
  - Deploy other GoldenGate built-in functions
  - Use multiple FILTERs on one statement
    - If any filter fails, the entire filter clause fails
  - Include multiple option clauses, for example (on insert/ update)
  - Raise a user-defined Error for exception processing

When multiple filters are specified per a given TABLE or MAP entry, the filters are executed until one fails or until all are passed. The failure of any filter results in a failure for all filters.

Filters can be qualified with operation type so you can specify different filters for inserts, updates and deletes.

The FILTER RAISEERROR option creates a user-defined error number if the filter clause is true. In the following example error 9999 is generated when the BEFORE timestamp is earlier than the CHECK timestamp. This also selects only update operations.

### 9.2.2.2 FILTER SYNTAX

Syntax

```
FILTER (<option> [, <option>] ) ,  
[FILTER (<option> [, <option>] ) ] [, ...]
```

Where <option> is one of :

```
<column specification>  
<field conversion function>  
<ON INSERT | UPDATE | DELETE >  
<IGNORE INSERT | UPDATE | DELETE>  
<RAISEERROR <error number> >
```

### 9.2.2.3 FILTER examples

The following example includes rows where the price multiplied by the amount exceeds 10,000:

```
FILTER ((PRODUCT_PRICE*PRODUCT_AMOUNT)>10000);
```

The following example includes rows containing a string "JOE":

```
FILTER (@STRFIND(NAME, "JOE")>0);
```

Why is the example above not constructed like the one below? The filter below will fail!

```
FILTER(NAME = "JOE"); (Will fail)
```

## 9.2.3 RANGE clause

### 9.2.3.1 RANGE overview

- Divides workload into multiple, randomly distributed groups of data
- Guarantees the same row will always be processed by the same process
- Determines which group that range falls in by computing a hash against the primary key or user defined columns

#### ***Syntax***

```
@RANGE (<my range>, <total ranges>
```

```
[, <column> [, ...]])
```

@RANGE computes a hash value of all the columns specified, or if no columns are specified, the primary key columns of the source table. A remainder of the hash and the total number of ranges is compared with the ownership range to determine whether or not @RANGE determines true or false. Note that the total number of ranges will be adjusted internally to optimize even distribution across the number of ranges.

**Restriction:** @RANGE cannot be used if primary key updates are performed on the database.



### 9.2.3.2 RANGE Examples

- For transaction volume beyond the capacity of a single Replicat, the example below shows three Replicat groups, each processing one-third of the data
- Hashing each operation by primary key to a particular Replicat guarantees the original sequence of operations

Replicat #1

```
MAP SALES.ACCOUNT,  
TARGET SALES.ACCOUNT, FILTER (@RANGE (1,3));
```

Replicat #2

```
MAP SALES.ACCOUNT,  
TARGET SALES.ACCOUNT, FILTER (@RANGE (2,3));
```

Replicat #3

```
MAP SALES.ACCOUNT,  
TARGET SALES.ACCOUNT, FILTER (@RANGE (3,3));
```

Two tables, REP and ACCOUNT, related by REP\_ID, require three Replicats to handle the transaction volumes

By hashing the REP\_ID column, related rows will always be processed to the same Replicat

```
RMTTRAIL /ggs/dirdat/aa  
TABLE SALES.REP, FILTER (@RANGE (1,3));  
TABLE SALES.ACCOUNT, FILTER (@RANGE (1,3,REP_ID));
```

```
RMTTRAIL /ggs/dirdat/bb  
TABLE SALES.REP, FILTER (@RANGE (2,3));  
TABLE SALES.ACCOUNT, FILTER (@RANGE (2,3,REP_ID));
```

```
RMTTRAIL /ggs/dirdat/cc  
TABLE SALES.REP, FILTER (@RANGE (3,3));  
TABLE SALES.ACCOUNT, FILTER (@RANGE (3,3,REP_ID));
```

## 9.2.4 Column Mapping

### 9.2.4.1 Overview

- GoldenGate provides the capability to map columns from one table to another
- Data can be transformed between dissimilar database tables
- Using COLMAP to map target columns from your source columns
- GoldenGate automatically matches source to target column names with USEDEFAULTS
- Mapping can be applied either when extracting or replicating data

Syntax for the COLMAP clause:

```
MAP SOURCE.TABLE, TARGET TARGET.TABLE,  
COLMAP ( [USEDEFAULTS,  
<target field> = <source expression>  
[, <target field> = <source expression>]  
[, ...]  
];
```

### 9.2.4.2 Example

```
MAP HR.CONTACT, TARGET HR.PHONE,  
COLMAP (USEDEFAULTS,  
NAME = CUST_NAME,  
PHONE_NUMBER = @STRCAT( "'", AREA_CODE, "'",  
PH_PREFIX, "-", PH_NUMBER ) );
```

This example:

- Moves the HR.CONTACT CUST\_NAME column value to the HR.PHONE NAME column
- Concatenates the HR.CONTACT AREA\_CODE, PH\_PREFIX and PH\_NUMBER with quote and hyphen literals to derive the PHONE\_NUMBER column value
- Automatically maps other HR.CONTACT columns to the HR.PHONE columns that have the same name.

## Building History

This example uses special values to build history of operations data

```
INSERTALLRECORDS  
MAP SALES.ACCOUNT, TARGET REPORT.ACCTHISTORY,  
COLMAP (USEDEFAULTS,  
TRAN_TIME = @GETENV("GGHEADER","COMMITTIMESTAMP"),  
OP_TYPE = @GETENV("GGHEADER", "OPTYPE"),  
BEFORE_AFTER_IND = @GETENV("GGHEADER",  
"BEFOREAFTERINDICATOR"),  
);
```

INSERTALLRECORDS causes Replicat to insert every change operation made to a record as a new record in the database. The initial insert and subsequent updates and deletes are maintained as point-in-time snapshots.

COLMAP uses the @GETENV function to get historical data from the GoldenGate trail header – TRAN\_TIME picks up the commit timestamp for the date of the transaction; OP\_TYPE stores whether it is an insert, update, or delete operation; and BEFORE\_AFTER\_IND indicates whether it is storing a “before” or “after” image.

## 9.3 Data Transformation

- **Functions**
- **SQLEXEC**
- **Macro**
- **User token**
- **User exits**

### 9.3.1 Functions

- GoldenGate provides the capability to transform columns by using a set of built-in functions
- Transformation functions can be applied either for Extract or Replicat
- If you require more, you also have the ability to call your own logic through user exits

Using column conversion functions, you can:

- Perform string and number conversion
- Extract portions of strings or concatenate columns
- Compare strings or numbers
- Perform a variety of date mappings
- Use single or nested IF statements to evaluate numbers, strings, and other column values to determine the appropriate value and format for target columns
- Functions are identified with the @ prefix

#### 9.3.1.1 Examples

```
MAP SALES.ACCOUNT, TARGET REPORT.ACCOUNT,  
COLMAP ( USEDEFAULTS,  
TRANSACTION_DATE = @DATE ("YYYY-MM-DD",  
"YY", YEAR,  
"MM", MONTH,  
"DD", DAY),  
AREA_CODE = @STREXT (PHONE-NO, 1, 3),  
PHONE_PREFIX = @STREXT (PHONE-NO, 4, 6),  
PHONE_NUMBER = @STREXT (PHONE-NO, 7, 10) );
```

#### 9.3.1.2 Functions – Performing Tests on Column Values

<u>Function</u>	<u>Description</u>
CASE	Allows user to select a value depending on a series of value tests
EVAL	Allows a user to select a value depending on a series of independent tests
IF	Selects one of two values depending on whether a conditional statement returns TRUE or FALSE
COLSTAT	Returns whether a column value is missing, NULL or invalid
COLTEST	Tests whether a column value is present, missing, NULL or invalid
VALONEOF	Returns true if a column contains one of a list of values

IF Function – Example

What IF clause would you use to set the target column AMOUNT\_COL to AMT only if AMT is greater than zero, and otherwise return zero?

***AMOUNT\_COL = @IF (AMT > 0, AMT, 0)***

What IF clause would you use to set ORDER\_TOTAL to PRICE\*QUANTITY if both PRICE and QUANTITY are greater than zero, otherwise return zero?

***ORDER\_TOTAL = @IF (PRICE > 0 AND QUANTITY > 0, PRICE \* QUANTITY, 0)***

### 9.3.1.3 Functions – Working with Dates

<u>Function</u>	<u>Description</u>
DATE	Returns a date from a variety of sources in a variety of output formats
DATEDIFF	Returns the difference between two dates or times
DATENOW	Returns the current date and time

What DATE expression would you use to convert year, month and day columns into a date?

***date\_col = @DATE ("YYYY-MM-DD", "YY", date1\_yy, "MM", date1\_mm, "DD", date1\_dd)***

What DATE expression would you use to convert a numeric column stored as YYYYMMDDHHMISS to a Julian timestamp?

***julian\_ts\_col = @DATE ("JTS", "YYYYMMDDHHMISS", numeric\_date)***

#### 9.3.1.4 Functions – Working with Strings and Numbers

<u>Function</u>	<u>Description</u>
COMPUTE	Returns the result of an arithmetic expression
NUMBIN	Converts a binary string into a number
NUMSTR	Converts a string into a number
STRCAT	Concatenates two or more strings
STRCMP	Compares two strings to determine if they are equal, or if the first is less or greater than the second
STREQ	Tests to see if two strings are equal. Returns 1 for equal and 0 if not equal
STREXT	Extracts selected characters from a string
STRFIND	Finds the occurrence of a string within a string
STRLEN	Returns the length of a string

<u>Function</u>	<u>Description</u>
STRLTRIM	Trims leading spaces in a column
STRNCAT	Concatenates one or more strings up to a specified number of characters per string
STRNCMP	Compares two strings up to a certain number of characters
STRNUM	Converts a number into a string, with justification and zero-fill options
STRRTRIM	Trims trailing spaces in a column
STRSUB	Substitutes one string for another within a column
STRTRIM	Trims both leading and trailing spaces in a column
STRUP	Changes a string to uppercase

Examples:

STRCAT expression to concatenate the columns LASTNAME and FIRSTNAME, separated by a semicolon

```
NAME = @STRCAT (LASTNAME, ";", FIRSTNAME)
```

STRCAT expression to concatenate a country code, area code and local phone number into an international phone number with hyphens between the components

```
INTL_PHONE = @STRCAT (COUNTRY_CODE, "-",  
AREA_CODE, "-", LOCAL_PHONE)
```

STREXT expressions to split a long-distance phone number into three columns (area code, prefix, phone no)

```
AREA_CODE = @STREXT (PHONE, 1, 3),  
PREFIX = @STREXT (PHONE, 4, 6),  
PHONE_NO = @STREXT (PHONE, 7, 10)
```

## **9.3.2 SQLEXEC**

### **9.3.2.1 Overview**

- Extends GoldenGate capabilities by enabling Extract and Replicat to communicate with the application database through SQL queries or run stored procedures
- Extends data integration beyond what can be done with GoldenGate functions

The SQ LEXEC option enables both Extract and Replicat to communicate with the user's database, either via SQL queries or stored procedures. SQLEXEC can be used to interface with a virtually unlimited set of the functionality supported by the underlying database.

- Execute a stored procedure or SQL query using the SQLEXEC clause of the TABLE or MAP parameter
- Optionally extract output parameters from the stored procedure or SQL query as input to a FILTER or COLMAP clause using the @GETVAL function
- Use SQLEXEC at the root level (without input/output parameters) to call a stored procedure, run a SQL query or issue a database command

Before defining the SQLEXEC clause, a database logon must be established. This is done via the SOURCEDB or USERID parameter for Extract, and the TARGETDB or USERID parameter for Replicat.

When using SQLEXEC, a mapping between one or more input parameters and source columns or column functions must be supplied.

### 9.3.2.2 Examples

The following stored procedure performs a query to return a description given a code:

```
CREATE OR REPLACE PROCEDURE LOOKUP  
(CODE_PARAM IN VARCHAR2, DESC_PARAM OUT VARCHAR2)  
BEGIN  
SELECT DESC_COL INTO DESC_PARAM  
FROM LOOKUP_TABLE  
WHERE CODE_COL = CODE_PARAM;  
END;
```

The following parameter entry:

- Maps data from the ACCOUNT table to the NEWACCT table
- When processing any rows from ACCOUNT, Extract performs the LOOKUP stored procedure prior to executing the column map
- Maps values returned in desc\_param to the newacct\_val column using the @GETVAL function

```
MAP HR.ACCOUNT, TARGET HR.NEWACCT,  
SQLEXEC (spname lookup,  
params (code_param = account_code)),  
COLMAP (USEDEFAULTS, newacct_id = account_id,  
newacct_val = @GETVAL(lookup.desc_param));
```

The following example (for Oracle) performs a SQL query directly to return the description. @GETVAL is used to retrieve the return parameter:

```
MAP HR.ACCOUNT, TARGET HR.NEWACCT,  
SQLEXEC (id lookup,  
query "select desc_param from lookup_table  
where code_col = :code_param",  
params (code_param = account_code)),  
COLMAP (USEDEFAULTS, newacct_id = account_id,  
newacct_val = @GETVAL(lookup.desc_param));
```



When specifying a stored procedure or query that updates the database, you must supply the DBOPS option in the SQLEXEC clause. Doing so ensures that any database updates are committed to the database properly.

#### SQLEXEC - Syntax within a TABLE or MAP Statement

When the SQLEXEC parameter is used within a TABLE or MAP statement, the syntax is:

```
SQLEXEC (  
{ SPNAME <sp name> | QUERY "<sql query>" }  
[, ID <logical name>]  
{ PARAMS <param spec> | NOPARAMS}  
[, BEFOREFILTER | AFTERFILTER]  
[, DBOP]  
[, EXEC <frequency>]  
[, ALLPARAMS <option>]  
[, PARAMBUFSIZE <num bytes>]  
[, MAXVARCHARLEN <num bytes>]  
[, TRACE <option>]  
[, ERROR <action>]  
)
```

The SQLEXEC option is specified as an option in TABLE and MAP statements within EXTRACT and REPLICAT parameter files. Use either SPNAME (for stored procedure) or QUERY (for SQL query).

#### SQLEXEC - Syntax as a Standalone Statement

When a SQLEXEC parameter is used at the root level, the syntax is:

```
SQLEXEC  
{ "call <sp name> ()" | "<sql query>" | "<database command>" }  
[EVERY <n> {SECONDS | MINUTES | HOURS | DAYS}]  
[ONEXIT]
```

Examples:

```
SQLEXEC "call prc_job_count ()"  
SQLEXEC "select x from dual "  
SQLEXEC "call prc_job_count ()" EVERY 30 SECONDS  
SQLEXEC "call prc_job_count ()" ONEXIT  
SQLEXEC "SET TRIGGERS OFF"
```

SQLEXEC error handling

**ERROR <action>**

Requires one of the following arguments:

**IGNORE** Database error is ignored and processing continues.

**REPORT** Database error is written to a report.

**RAISE** Database error is handled just as a table replication error.

**FINAL** Database error is handled as a table replication error, but does not process any additional queries.

**FATAL** Database processing abends.

### **9.3.3 Macros**

#### **9.3.3.1 Overview**

Macros enable easier and more efficient building of parameters

- Write once and use many times
- Consolidate multiple statements
- Eliminate the need for redundant column specifications
- Use macros to invoke other macros
- Create macro libraries and share across parameter files

GoldenGate macros work with the following parameter files:

- Manager
- Extract
- Replicat

### 9.3.3.2 Macros – Creation

- Macros can be defined in any parameter file or library
- Macro statements include the following
  - Macro name
  - Optional parameter list
  - Macro body
- Syntax

```
MACRO #<macro name>
```

```
PARAMS (#<param1>, #<param2>, ...)
```

```
BEGIN
```

```
<macro body>
```

```
END;
```

- The macro and parameter identifier ‘#’ can be changed to alternative value

```
MACROCHAR $
```

### 9.3.3.3 Macros – Invoking

Reference the macro and parameters anywhere you want the macro to be invoked

```
EXTRACT EXSALES
```

```
MACRO #make_date
```

```
PARAMS (#year, #month, #day)
```

```
BEGIN
```

```
@DATE("YYYY-MM-DD", "CC", @IF(#year < 50, 20, 19),
```

```
"YY", #year, "MM", #month, "DD", #day)
```

```
END;
```

```
MAP SALES.ACCT, TARGET REPORT.ACCOUNT,
```

```
COLMAP
```

```
(
```

```
TARGETCOL1 = SOURCECOL1,
```

```
Order_Date = #make_date(Order_YR, Order_MO, Order_DAY),
```

```
Ship_Date = #make_date(Ship_YR, Ship_MO, Ship_DAY)
```

```
);
```

#### 9.3.3.4 Macros – Example

Consolidating Multiple Parameters

Define the macro:

```
MACRO #option_defaults  
BEGIN  
GETINSERTS  
GETUPDATES  
GETDELETES  
INSERTDELETES  
END;
```

Invoke the macro:

```
#option_defaults ()  
IGNOREUPDATES  
MAP SALES.SRCTAB, TARGET SALES.TARGETAB;  
#option_defaults ()  
MAP SALES.SRCTAB2, TARGET SALES.TARGETAB2;
```

The macro expands to the following:

```
GETINSERTS  
GETUPDATES  
GETDELETES  
INSERTDELETES  
IGNOREUPDATES  
MAP SALES.SRCTAB, TARGET SALES.TARGETAB;  
GETINSERTS  
GETUPDATES  
GETDELETES  
INSERTDELETES  
MAP SALES.SRCTAB2, TARGET SALES.TARGETAB2;
```

### 9.3.3.5 Macros – Libraries

Macros can be built in a library and referenced into your parameter file

```
EXTRACT EXTACCT  
INCLUDE /ggs/dirprm/macro.lib
```

### 9.3.4 User Tokens

GoldenGate provides the ability to store environmental values in the user token area of the GoldenGate record header.

Set token values through a TABLE TOKENS clause and @GETENV functions, for example:

```
TABLE SALES.PRODUCT,  
TOKENS (TKN1 = @GETENV("GGENVIRONMENT", "OSUSERNAME"),  
TKN2 = @GETENV("GGHEADER", "COMMITTIMESTAMP") );
```

Use token values to populate target columns through a MAP COLMAP clause and @TOKEN functions, for example:

```
MAP SALES.PRODUCT, TARGET SALES.PRODUCT_HISTORY,  
COLMAP (USEDEFAULTS, OSUSER = @TOKEN("TKN1"),  
TRANSTIME = @TOKEN("TKN2") );
```

#### Saving user tokens

Use the TOKENS option of the Extract TABLE parameter to define a user token and associate it with data. Tokens enable you to extract and store data within the user token area of a trail record header. You can set tokens to values returned by the @GETENV function (for example, values from the GoldenGate header or environment).

#### Using user tokens

You can use token data in column maps, stored procedures called by SQLEXEC, or macros. For example, use the @TOKEN function in the COLMAP clause of a Replicat MAP statement to map a token to a target column.

#### 9.3.4.1 Environmental Values Available to @GETENV

Syntax: @GETENV ("<option>", ["<return value>"])

Example: @GETENV ("GGENVIRONMENT", "HOSTNAME")

Source	Option	Returns values for/from
General	"LAG"	Lag (in unit specified)
	"LASTERR"	Last failed operation
	"JULIANTIMESTAMP"	Julian timestamp
	"RECSOUTPUT"	Number of records written to trail
GoldenGate	"GGENVIRONMENT"	GoldenGate environment
	"GGFILEHEADER"	Trail file header
	"GGHEADER"	Trail record header
	"RECORD"	Trail record location
Database	"DBENVIRONMENT"	Database environment
	"TRANSACTION"	Source transaction
Operating System	"OSVARIABLE"	OS environmental variable

#### 9.3.4.1.1 General:

**@GETENV ("LAG", "<unit>")**

"SEC" Returns the lag in seconds. This is the default when a unit is not explicitly provided for LAG.

"MSEC" Returns the lag in milliseconds.

"MIN" Returns the lag in minutes.

**@GETENV ("LASTERR", "<return value>")**

"DBERRNUM" Returns the database error number associated with the failed operation.

"DBERRMSG" Returns the database error message associated with the failed operation.

"OPTYPE" Returns the operation type that was attempted.

"ERRTYPE" Returns the type of error: DB (for database errors) or MAP (for mapping errors).

**@GETENV ("JULIANTIMESTAMP")**

**@GETENV ("RECSOUTPUT")**

#### 9.3.4.1.2 Golden Gate:

**@GETENV ("GGENVIRONMENT", "<return value>")**

"DOMAINNAME" (Windows only) Returns the domain name associated with the user that started the process.

"GROUPDESCRIPTION" The description of the group (if any) taken from the checkpoint file.

"GROUPNAME" Returns the name of the process group.

"GROUPTYPE" Returns the type of process, either EXTRACT or REPLICAT.

“HOSTNAME” Returns the name of the system running the Extract or Replicat process.  
“OSUSERNAME” Returns the operating system user name that started the process.  
“PROCESSID” The process ID that is assigned by the operating system.

**@GETENV (“GGHEADER”, “<return value>”)**

“BEFOREAFTERINDICATOR” Returns BEFORE (before image) or AFTER (after image).  
“COMMITTIMESTAMP” Returns the transaction timestamp (when committed) in the format of YYYY-MM-DD HH:MI:SS.FFFFFFFF.  
“LOGPOSITION” Returns the sequence number in the data source.  
“LOGRBA” Returns the relative byte address in the data source.  
“OBJECTNAME” | “TABLENAME” Returns the table name or object name (if a sequence).  
“OPTYPE” Returns the type of operation: INSERT, UPDATE, DELETE, ENSCRIBECOMPUPDATE, SQL COMPUPDATE, PK UPDATE, TRUNCATE, TYPE n.  
“RECORDLENGTH” Returns the record length in bytes.  
“TRANSACTIONINDICATOR” Returns the transaction indicator: BEGIN, MIDDLE, END, WHOLE.

**@GETENV (“GGFILEHEADER”, “<return\_value>”)**

“COMPATIBILITY” Returns the GoldenGate compatibility level of the trail file.

- 1 means that the trail file is of GoldenGate version 10.0 or later, which supports file headers that contain file versioning information.
- 0 means that the trail file is of a GoldenGate version that is older than 10.0. File headers are not supported in those releases. The 0 value is used for backward compatibility to those GoldenGate versions.

***Information about the trail file***

“CHARSET” Returns the global character set of the trail file. For example:  
WCP1252-1  
“CREATETIMESTAMP” Returns the time that the trail was created, in local GMT Julian time in INT64.  
“FILENAME” Returns the name of the trail file. Can be an absolute or relative path.  
“FILEISTRAIL” Returns a True/False flag indicating whether the trail file is a single file (such as one created for a batch run) or a sequentially numbered file that is part of a trail for online, continuous processing.  
“FILESEQNO” Returns the sequence number of the trail file, without any leading zeros.  
“FILESIZE” Returns the size of the trail file when the file is full and the trail rolls over.  
“FIRSTRECCSN” Returns the commit sequence number (CSN) of the first record in the trail file. NULL until the trail file is completed.  
“LASTRECCSN” Returns the commit sequence number (CSN) of the last record in the trail file. NULL until the trail file is completed.

“FIRSTRECIOTIME” Returns the time that the first record was written to the trail file. NULL until the trail file is completed.

“LASTRECIOTIME” Returns the time that the last record was written to the trail file. NULL until the trail file is completed.

“URI” Returns the universal resource identifier of the process that created the trail file, in the format: <host\_name>:<dir>[:<dir>][:<dir\_n>]<group\_name>

“URIHISTORY” Returns a list of the URIs of processes that wrote to the trail file before the current process.

***Information about the GoldenGate process that created the trail file***

“GROUPNAME” Returns the group name associated with the Extract process that created the trail. The group name is that which was given in the ADD EXTRACT command. For example, “ggext.”

“DATASOURCE” Returns the data source that was read by the process.

“GGMAJORVERSION” Returns the major version of the Extract process that created the trail, expressed as an integer. For example, if a version is 1.2.3, it returns 1.

“GGMINORVERSION” Returns the minor version of the Extract process that created the trail, expressed as an integer. For example, if a version is 1.2.3, it returns 2.

“GGVERSIONSTRING” Returns the maintenance (or patch) level of the Extract process that created the trail, expressed as an integer. For example, if a version is 1.2.3, it returns 3.

“GGMAINTENANCELEVEL” Returns the maintenance version of the process (xx.xx.xx).

“GGBUGFIXLEVEL” Returns the patch version of the process (xx.xx.xx.xx).

“GGBUILDNUMBER” Returns the build number of the process.

***Information about the local host of the trail file***

“HOSTNAME” Returns the DNS name of the machine where the Extract that wrote the trail is running.

“OSVERSION” Returns the major version of the operating system of the machine where the Extract that wrote the trail is running.

“OSRELEASE” Returns the release version of the operating system of the machine where the Extract that wrote the trail is running.

“OSTYPE” Returns the type of operating system of the machine where the Extract that wrote the trail is running.

“HARDWARETYPE” Returns the type of hardware of the machine where the Extract that wrote the trail is running.

***Information about the database that produced the data in the trail file.***

“DBNAME” Returns the name of the database, for example findb.

“DBINSTANCE” Returns the name of the database instance, if applicable to the database type, for example ORA1022A.

“DBTYPE” Returns the type of database that produced the data in the trail file.

“DBCHARSET” Returns the character set that is used by the database that produced the data in the trail file.



“DBMAJORVERSION” Returns the major version of the database that produced the data in the trail file.

“DBMINORVERSION” Returns the minor version of the database that produced the data in the trail file.

“DBVERSIONSTRING” Returns the maintenance (patch) level of the database that produced the data in the trail file.

“DBCLIENTCHARSET” Returns the character set that is used by the database client.

“DBCLIENTVERSIONSTRING” Returns the maintenance (patch) level of the database client.

***Recovery information carried over from the previous trail file***

“RECOVERYMODE” Returns recovery information for internal GoldenGate use.

“LASTCOMPLETECSN” Returns recovery information for internal GoldenGate use.

“LASTCOMPLETEXIDS” Returns recovery information for internal GoldenGate use.

“LASTCSN” Returns recovery information for internal GoldenGate use.

“LASTXID” Returns recovery information for internal GoldenGate use.

“LASTCSNTS” Returns recovery information for internal GoldenGate use.

**@GETENV (“RECORD”, “<environment value>”)**

“FILESEQNO” Returns the sequence number of the trail file without any leading zeros.

“FILERBA” Returns the relative byte address of the record within the FILESEQNO file.

**9.3.4.1.3 Database:**

**@GETENV (“DBENVIRONMENT”, “<return value>”)**

“DBNAME” Returns the database name.

“DBVERSION” Returns the database version.

“DBUSER” Returns the database login user.

“SERVERNAME” Returns the name of the server.

**@GETENV (“TRANSACTION”, “<return value>”)**

“TRANSACTIONID” | “XID” Returns the transaction ID number.

“CSN” Returns the commit sequence number (CSN).

“TIMESTAMP” Returns the commit timestamp of the transaction.

“NAME” Returns the transaction name, if available.

“USERID” (Oracle) Returns the Oracle user-id of the database user that committed the last transaction.

“USERNAME” (Oracle) Returns the Oracle user-name of the database user that committed the last transaction.

“RSN” Returns the record sequence number.

“PLANNAME” (DB2 on z/OS) Returns the plan name under which the current transaction was originally executed. The plan name is included in the begin unit of recovery log record.

#### 9.3.4.1.4 OS Variable:

**@GETENV ("OSVARIABLE", "<variable>")**

"<variable>" The name of the variable. The search is an exact match of the supplied variable name. The search is case-sensitive if the operating system supports case-sensitivity.

#### 9.3.4.2 Example

##### Setting:

```
EXTRACT EXTDEMO  
TABLE SALES.PRODUCT, TOKENS (  
TKN-OSUSER = @GETENV ("GGENVIRONMENT", "OSUSERNAME"),  
TKN-DOMAIN = @GETENV ("GGENVIRONMENT", "DOMAINNAME"),  
TKN-COMMIT-TS = @GETENV ("GGHEADER", "COMMITTIMESTAMP"),  
TKN-BA-IND = @GETENV ("GGHEADER", "BEFOREAFTERINDICATOR),  
TKN-TABLE = @GETENV ("GGHEADER", "TABLENAME"),  
TKN-OP-TYPE = @GETENV ("GGHEADER", "OPTYPE"),  
TKN-LENGTH = @GETENV ("GGHEADER", "RECORDLENGTH"),  
TKN-DB-VER = @GETENV ("DBENVIRONMENT", "DBVERSION");
```

##### Using:

Tokens are retrieved through a MAP COLMAP clause and @TOKEN functions:

```
MAP SALES.ORDER, TARGET REPORT.ORDER_HISTORY,  
COLMAP (USEDEFAULTS,  
TKN_NUMRECS = @TOKEN ("TKN-NUMRECS");  
MAP SALES.CUSTOMER, TARGET REPORT.CUSTOMER_HISTORY,  
COLMAP (USEDEFAULTS,  
TRAN_TIME = @TOKEN ("TKN-COMMIT-TS"),  
OP_TYPE = @TOKEN ("TKN-OP-TYPE"),  
BEFORE_AFTER_IND = @TOKEN ("TKN-BA-IND"),  
TKN_ROWID = @TOKEN ("TKN-ROWID"));
```

## **9.3.5 User Exits**

### **9.3.5.1 Overview**

- Custom logic written in C or C++ by the customer
- Invoked at different points in Extract or Replicat processing (through CUSEREXIT parameter)
- Allows you to extend or customize the functionality of data movement and integration beyond what is supported through mapping, functions, or SQLEXEC
- Can perform an unlimited number of functions

### **9.3.5.2 User Exits – Applications**

- Perform arithmetic operations or data transformations beyond those provided with GoldenGate built-in functions
- Perform additional table lookups or clean up invalid data
- Respond to events in custom ways, for example, by sending a formatted e-mail message or paging a supervisor based on some field value
- Accumulate totals and gather statistics
- Perform conflict detection, or custom handling of errors or discards
- Determine the net difference in a record before and after an update (conflict resolution technique)

### **9.3.5.3 Parameters**

- EXIT\_CALL\_TYPE indicates when, during processing, the Extract or Replicat process calls the user exit: at start processing, stop processing, begin transaction, end transaction, process record, process marker, discard record, fatal error or call result
- EXIT\_CALL\_RESULT provides a response to the routine: OK, ignore, stop, abend or skip record
- EXIT\_PARAMS supplies information to the routine: calling program path and name, function parameter, “more records” indicator
- ERCALLBACK implements a callback routine. Callback routines retrieve record and GoldenGate context information and modify the contents of data records

#### 9.3.5.4 Implementing

- On Windows: create a DLL in C and create a routine to be called from Extract or Replicat
- On UNIX: Create a shared object in C and create a routine to be called from Extract or Replicat
- The routine must accept the following parameters:
  - **EXIT\_CALL\_TYPE**
  - **EXIT\_CALL\_RESULT**
  - **EXIT\_PARAMS**
- In the source for the DLL/shared object, include the "usrdecs.h" file (in the GoldenGate install directory)
- Call the ERCALLBACK function from the shared object to retrieve record and application context information

#### 9.3.5.5 Calling

- You can call a user exit from Extract or Replicat by the CUSEREXIT parameter
- Syntax

***CUSEREXIT <DLL or shared object name> <routine name>***

***[, PASSTHRU]***

***[, INCLUDEUPDATEBEFORES]***

***[, PARAMS "<startup string>"]***

- Examples

***CUSEREXIT userexit.dll MyUserExit***

***CUSEREXIT userexit.dll MyUserExit, INCLUDEUPDATEBEFORES, &***

***PASSTHRU, PARAMS "init.properties"***

<DLL or shared object name>

The name of the Windows DLL or UNIX shared object that contains the user exit function.

<routine name>

The name of the exit routine to be executed.

PASSTHRU

Valid only for an Extract data pump. It assumes that no database is required, and that no output trail is allowed. It expects that the user exit will perform all of the processing and that Extract will skip the record.

Extract will perform all of the required data mapping before passing the record to the user exit. Instead of a reply status of EXIT\_OK\_VAL, the reply will be EXIT\_PROCESSED\_REC\_VAL. All process statistics are updated as if the records were processed by GoldenGate.

INCLUDEUPDATEBEFORES

Passes the before images of column values to a user exit. When using this parameter, you must explicitly request the before image by setting the requesting\_before\_after\_ind flag to BEFORE\_IMAGE\_VAL within a callback function that supports this flag. Otherwise, only the after image is passed to the user exit. By default, GoldenGate only works with after images.

When using INCLUDEUPDATEBEFORES for a user exit that is called from a data pump or from Replicat, always use the GETUPDATEBEFORES parameter for the primary Extract process, so that the before image is captured, written to the trail, and causes a process record event in the user exit. In a case where the primary Extract also has a user exit, GETUPDATEBEFORES causes both the before image and the after image to be sent to the user exit as separate EXIT\_CALL\_PROCESS\_RECORD events.

If the user exit is called from a primary Extract (one that reads the transaction log), only INCLUDEUPDATEBEFORES is needed for that Extract. GETUPDATEBEFORES is not needed in this case, unless other GoldenGate processes downstream will need the before image to be written to the trail. INCLUDEUPDATEBEFORES does not cause before images to be written to the trail.

PARAMS "<startup string>"

Passes the specified string at startup. Can be used to pass a properties file, startup parameters, or other string. The string must be enclosed within double quote marks.

Data in the string is passed to the user exit in the EXIT\_CALL\_START exit\_params\_def.function\_param. If no quoted string is specified with PARAMS, the exit\_params\_def.function\_param is NULL.

### 9.3.6 Oracle sequences

- GoldenGate supports the replication of Oracle sequence values
- Use the Extract SEQUENCE parameter to extract sequence values from the transaction log; for example:

***SEQUENCE hr.employees\_seq;***

- Use the Replicat MAP parameter to apply sequence values to the target;

For example:

***MAP hr.employees\_seq, TARGET payroll.employees\_seq;***

- The default Replicat CHECKSEQUENCEVALUE parameter ensures that target sequence values are:

Higher than the source values (if the increment interval is positive) or

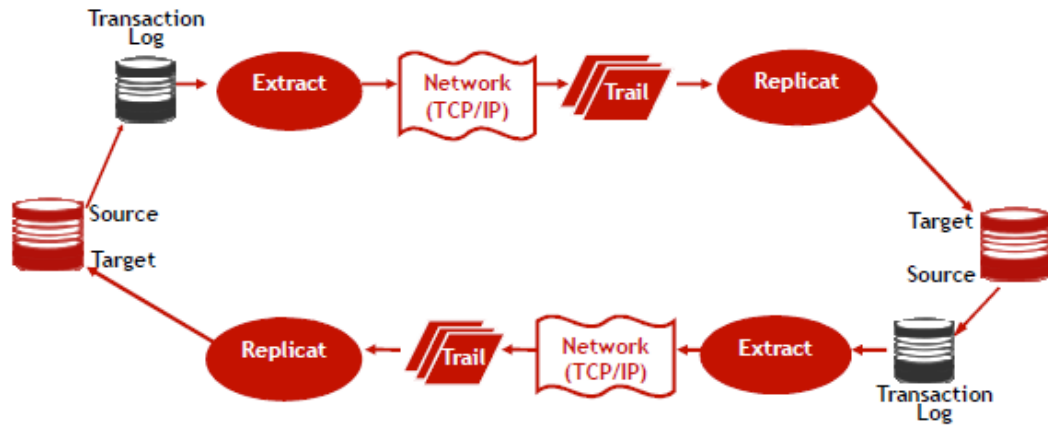
Lower than the source values (if the increment interval is negative)

**Golden Gate supports online and batch change synchronization methods for oracle sequences**

**Golden Gate does not supports oracle sequence replication for initial loads and bi-directional replication.**

# 10. Bi-directional Replication

## 10.1 Overview



- Available for both homogeneous and heterogeneous configurations
- Distributed processing
- Both sides are live
- GoldenGate's low latency reduces the risk of conflicts
- GoldenGate provides loop detection

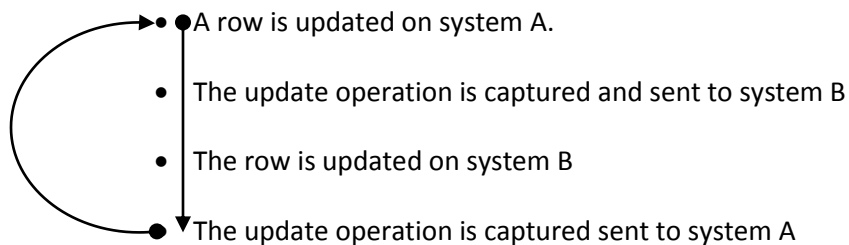
Configuring GoldenGate for bidirectional may be as straightforward as configuring a mirror set of Extract and Replicat groups moving in the opposite direction.

## 10.2 Bidirectional issues

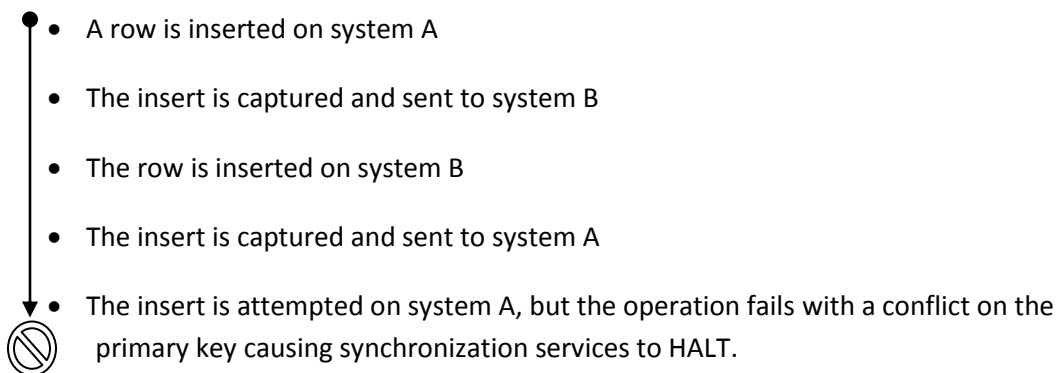
- Loop Detection
- Conflict Avoidance, Detection and Resolution
- Truncate table operations

### 10.2.1 Loop detection

#### Endless Loop



#### Service halt due to loop creation



#### 10.2.1.1 Loop prevention

To avoid looping, Replicat's operations must be prevented from being sent back to the source table by Extract. This is accomplished by configuring Extract to ignore transactions issued by the Replicat user.

Loop detection technique depends on the source database:



## Oracle

Using EXCLUDEUSER or EXCLUDEUSERID (Oracle 10g and later)

Stop capture with Extract parameter:

- **TRANLOGOPTIONS EXCLUDEUSER <userid | username>**
- Use **@GETENV ("TRANSACTION", "USERID")** or **@GETENV ("TRANSACTION", "USERNAME")** to retrieve the Oracle userid or username on the Replicat database

Using Trace Table (Oracle 9i and earlier)

Add a trace table to detect GoldenGate operations with the GGSCI command:

**ADD TRACETABLE <owner>.<table name>**

The default <table name> is **GGG\_TRACE**

If not using the default table name, add a parameter in both Extract and Replicat:

**TRACETABLE <owner>.<table name>**

## SQL Server

By default, SQL Server does not log the ggs\_repl transactions written by Replicat, so no loops arise.

## DB2 and Ingres

Execute Replicat with unique User ID

Stop capture with Extract parameter:

**TRANLOGOPTIONS EXCLUDEUSER <user id>**

## Sybase

Do nothing and allow Replicat to use the default transaction name ggs\_repl

Or identify the Replicat transaction name in the Extract parameter:

**TRANLOGOPTIONS EXCLUDETRANS <trans name>**

Or identify the Replicat user name in the Extract parameter:

**TRANLOGOPTIONS EXCLUDEUSER <user name>**

## NonStop SQL/MX

Replicat transactions identified by the name of the checkpoint table specified with TRANLOGOPTIONS option:

***TRANLOGOPTIONS FILTERTABLE <table>***

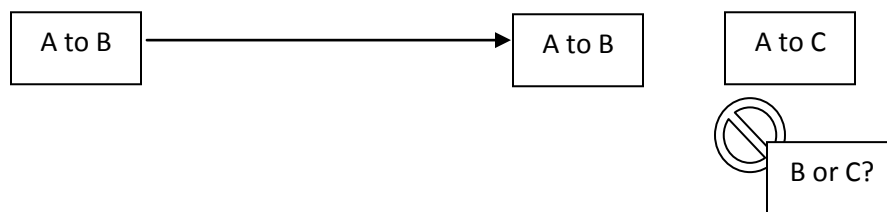
Extract ignores transactions that include this checkpoint table

PURGEDATA operation is not supported

### **Teradata**

You do not need to identify Replicat transactions that are applied to a Teradata database.

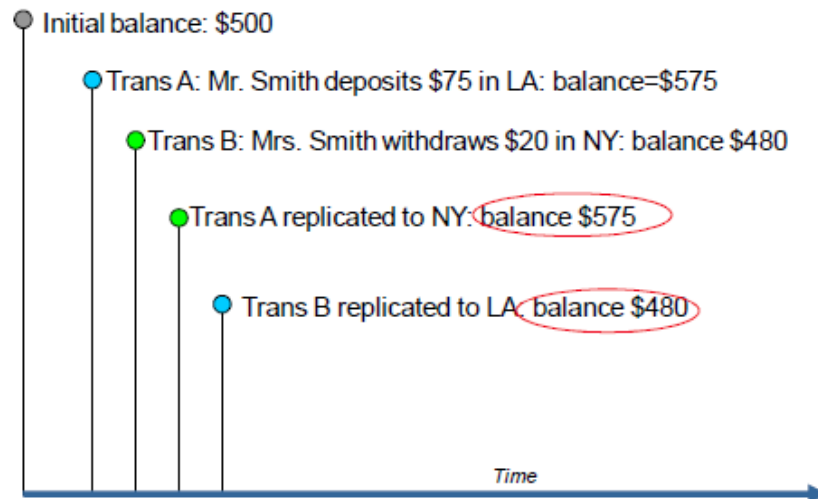
## **10.2.2 Conflicts**



### **10.2.2.1 Resolving Conflicts**

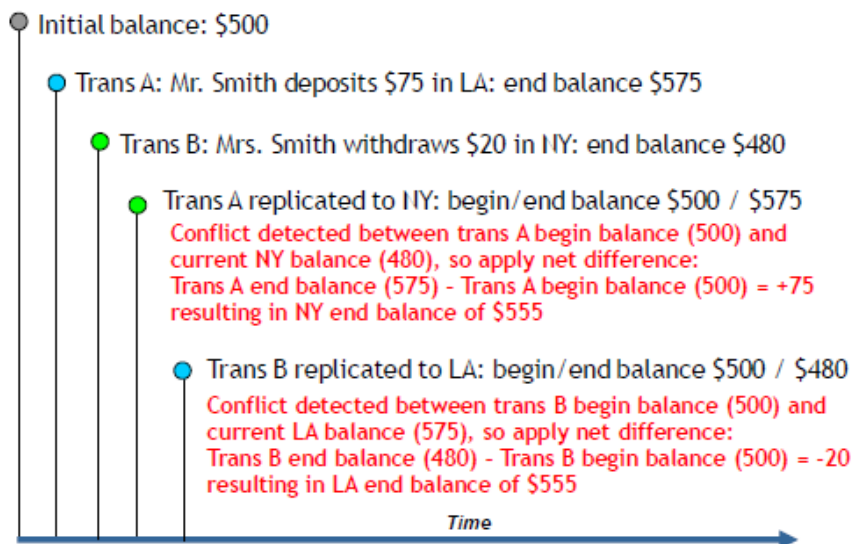
1. Conflicts can be minimized by low latency
2. Conflicts can be avoided at the application level by assuring that records are always updated on one system only
3. GoldenGate can capture both the before and after image of updates so you can compare before applying updates
  - 3.1 Apply the net difference instead of the after value
  - 3.2 Map the data to an exception table for manual resolution
  - 3.3 Accept or ignore the change based on date/time

**Example 1:**



At end of day, correct balance should be  $\$500 + \$75 - \$20 = \$555!$

**Solution is sending the difference.**



### Example 2:

Conflict detection and resolution using SQLEXEC

```
REPERROR 9999, EXCEPTION
MAP SRCTAB, TARGET TARGTAB,
SQLEXEC (ID CHECK, ON UPDATE, BEFOREFILTER,
QUERY "SELECT COUNTER FROM TARGTAB"
"WHERE PKCOL = :P1",
PARAMS (P1 = PKCOL)),
FILTER (ON UPDATE, BEFORE.COUNTER <> CHECK.COUNTER,
RAISEERROR 9999);
INSERTALLRECORDS
MAP SRCTAB, TARGET TARGET_EXCEPT, EXCEPTIONSONLY,
COLMAP (USEDEFAULTS, ERRTYPE = "Conflict Detected");
```

The example above does the following when an update is encountered:

- Before the update filter is executed, perform a query to retrieve the present value of the COUNTER column
- If the first filter is passed successfully, ensure the value of COUNTER before the update occurred matches the value in the target before performing the update.
- If the update filter fails, raise error 9999
- The REPERROR clause for error 9999 ensures that the exceptions map to TARGET\_EXCEPT will be executed

## **10.3 Truncate Table operation**

- Truncate Table operations cannot be detected for loops
- Make sure that GETTRUNCATES is ON for only one direction
- Use IGNORETRUNCATES (default) for the other direction
- Change database security so truncates can only be issued on one system

# 11.DDL Replication

## 11.1 Overview

- Available for all Oracle versions supported by GoldenGate DML synchronization
- DDL in relation to DML:
  - DDL can be active with/without DML synchronization
  - The same Extract/Replicat should be processing both DML and DDL to avoid timing issues
- DDL operations are recognized differently by Extract and Replicat:
  - Source: DDL disabled by default; Extract must be configured to enable.
  - Target: DDL enabled by default to maintain data integrity
  - Replicat must be configured to ignore or filter DDL

## 11.2 Requirements/Restrictions

- Identical source and target data def: ASSUMETARGETDEFS
- Data pumps must be configured in PASSTHRU mode
- Data manipulation is not supported
- WILDCARDRESOLVE must remain set to DYNAMIC (default)
- GoldenGate user exits are not supported for DDL activity
- Restrictions exist for DDL operations that involve user defined types and LOB data
- GoldenGate does not support bidirectional replication of DDL – allow DDL changes on one system only
- DDL statements > 2MB require special handling
- The GETTRUNCATES parameter should not be used with full DDL support
- Table name cannot be longer than 16 characters plus quotation marks for ALTER TABLE RENAME
- ALTER TABLE..MOVE TABLESPACE

- Supported when tablespace all SMALLFILE or BIGFILE
- Stop Extract before issuing a MOVE TABLESPACE
- The recycle bin must be turned off
- ALTER DATABASE and ALTER SYSTEM not captured
- Long DDL statements need special handling

#### **Supported objects for Oracle**

Clusters, Tables, Functions, Tablespaces, indexes, triggers, packages, types, procedures  
views, roles, materialized views, sequences, users, synonyms

### **11.3 Configuring DDL Replication**

1. Run the scripts as SYSDBA from GG Installation Home

```
SQL> @marker_setup
```

2. Provide inputs when asked

```
Enter GoldenGate schema name: OGG
```

3. Turn recyclebin off

```
SQL> alter session set recyclebin=OFF;
```

4. Run the DDL setup script

```
SQL> @ddl_setup
```

```
Enter GoldenGate schema name: OGG
```

5. Run the Role setup script to grant GGS\_GGSUSER\_ROLE to the GG owner user in the database.

```
SQL> @role_setup
```

```
Enter GoldenGate schema name: OGG
```

6. Grant GGS\_GGSUSER\_ROLE to GG owner user in database

```
SQL> grant ggs_ggsuser_role to OGG;
```

7. Run the script to enable DDL trigger

```
SQL> @ddl_enable
```

8. Run script to pin tracing for performance evaluation

```
SQL> @ddl_pin GGS_OWNER
```

**Following objects are created when DDL configuration is done:**

- DDL marker table: GGS\_MARKER
  - Stores DDL information
- DDL history table: GGS\_DDL\_HIST
  - Stores object metadata history
- DDL trigger: GGS\_DDL\_TRIGGER\_BEFORE
  - Activates when there is a DDL operation
  - Writes to the Marker and History table
- User Role: GGS\_GGSUSER\_ROLE
  - Establishes the role needed for DDL replication
  - Should be the user that executes Extract

## **11.4 DDL Replication Scope**

### **Mapped**

- Objects that are specified in TABLE and MAP statements
- Operations CREATE, ALTER, DROP, RENAME, GRANT\*, REVOKE\*
- Objects TABLE\*, INDEX, TRIGGER, SEQUENCE\*, MATERIALIZED VIEW\*
- Operations are only for objects with asterisk

## Unmapped

Objects that are not specified in TABLE and MAP statements

## Other scope

DDL operations that cannot be mapped are of OTHER scope, just like the follow operations: create user, grant role, create tablespace, alter datafile

### Example:

```
ddl &
include mapped exclude objtype 'table' objname "hr.jobs", &
exclude INSTRWORDS 'alter add "city"', &
include unmapped, objname "hr3.jobs", &
include other
```

## 11.5 Syntax

Valid for Extract and Replicat

```
DDL [
  {INCLUDE | EXCLUDE}
  [, MAPPED | UNMAPPED | OTHER | ALL]
  [, OPTYPE <type>]
  [, OBJTYPE '<type>']
  [, OBJNAME "<name>"]
  [, INSTR '<string>']
  [, INSTRCOMMENTS '<comment_string>']
]
[...]
```

All criteria specified with multiple options must be satisfied for a DDL statement to be replicated.

### Options

INCLUDE | EXCLUDE Identifies the beginning of an *inclusion* or *exclusion* clause.

INCLUDE includes specified DDL for capture or replication. EXCLUDE excludes specified DDL from being captured or replicated.

The inclusion or exclusion clause must consist of the INCLUDE or EXCLUDE keyword followed by any valid combination of other options of the DDL parameter.

An EXCLUDE must be accompanied by a corresponding INCLUDE clause. An EXCLUDE takes priority over any INCLUDEs that contain the same criteria. You can use multiple inclusion and exclusion clauses.



MAPPED | UNMAPPED | OTHER | ALL applies INCLUDE or EXCLUDE based on the DDL operation scope.

- MAPPED applies to DDL operations that are of MAPPED scope.
- UNMAPPED applies to DDL operations that are of UNMAPPED scope.
- OTHER applies to DDL operations that are of OTHER scope.
- ALL applies to DDL operations of all scopes. DDL EXCLUDE ALL maintains up-to-date metadata on objects, while blocking the replication of the DDL operations themselves.

OPTYPE <type> applies INCLUDE or EXCLUDE to a specific type of DDL operation.

For <type>, use any DDL command that is valid for the database, such as CREATE, ALTER and RENAME.

OBJTYPE '<type>' applies INCLUDE or EXCLUDE to a specific type of database object. For <type>, use any object type that is valid for the database, such as TABLE, INDEX, TRIGGER, USER, ROLE. Enclose the object type within single quotes.

OBJNAME "<name>" applies INCLUDE or EXCLUDE to the name of an object, for example a table name. Provide a double-quoted string as input. Wildcards can be used. If you do not qualify the object name for Oracle, the owner is assumed to be the GoldenGate user. When using OBJNAME with MAPPED in a Replicat parameter file, the value for OBJNAME must refer to the name specified with the TARGET clause of the MAP statement. For DDL that creates triggers and indexes, the value for OBJNAME must be the name of the base object, not the name of the trigger or index. For RENAME operations, the value for OBJNAME must be the new table name.

INSTR '<string>' applies INCLUDE or EXCLUDE to DDL statements that contain a specific character string within the command syntax itself, but not within comments.

Enclose the string within single quotes. The string search is not case sensitive

INSTRCOMMENTS '<comment\_string>'s applies INCLUDE or EXCLUDE to DDL statements that contain a specific character string within a comment, but not within the DDL command itself. By using INSTRCOMMENTS, you can use comments as a filtering agent. Enclose the string within single quotes. The string search is not case sensitive. You can combine INSTR and INSTRCOMMENTS options to filter on a string in the command syntax and in the comments.

## 11.6 String Substitution

- DDLSUBST parameter substitutes strings in a DDL operation
- Multiple statements can be used
- DDLSUBST parameter syntax:

```
DDLSUBST '<search_string>' WITH '<replace_string>'
```

```
[INCLUDE <clause> | EXCLUDE <clause>]
```

Where:

'<search\_string>' is the string in the source DDL statement you want to replace, in single quotes

'<replace\_string>' is the replacement string, in single quotes

<clause> is an inclusion or exclusion clause using same syntax as INCLUDE and EXCLUDE from DDL parameter

**Example:**

DDLSUBST 'users' with 'system' include mapped objtype 'table' INSTRWORDS 'create table tablespace "users"'

## 11.7 DDL Options

DDLOPTIONS parameter configures aspects of DDL processing other than filtering and string substitution

***DDLOPTIONS***

***[, MAPDERIVED | NOMAPDERIVED]***

***[, NOCROSSRENAME]***

***[, REPORT | NOREPORT]***

***[, ADDTRANDATA]***

***[, DEFAULTUSERPASSWORD <password>***

***[ENCRYPTKEY DEFAULT | ENCRYPTKEY <keyname>]]***

***[, GETAPPLOPS | IGNOREAPPLOPS]***

***[, GETREPLICATES | IGNOREREPLICATES]***

***[, REMOVECOMMENTS {BEFORE | AFTER}]***

***[, REPLICATEPASSWORD | NOREPLICATEPASSWORD]***

MAPDERIVED | NOMAPDERIVED is valid for Replicat. It controls how derived Object (e.g. indexes) names are mapped. With MAPDERIVED, if a MAP statement exists for the derived object that is used. Otherwise, the name is mapped to the name specified in the TARGET clause of the MAP statement for the base object. MAPDERIVED is the default. NOMAPDERIVED overrides any explicit MAP statements that contain the name of the derived object and prevents name mapping. NOCROSSRENAME is valid for Extract on Oracle RAC. It assumes that tables excluded from the GoldenGate configuration will not be renamed to names that are in the configuration. NOCROSSRENAME improves performance by eliminating processing that otherwise is required to keep track of excluded tables in case they get renamed to an included name.

REPORT | NOREPORT is valid for Extract and Replicat. It controls whether or not expanded DDL processing information is written to the report file. The default of NOREPORT reports basic DDL statistics. REPORT adds the parameters being used and a step-by-step history of the operations that were processed

ADDTRANDATA is valid for Extract. Use ADDTRANDATA to enable supplemental logging for CREATE TABLE or update supplemental logging for tables affected by an ALTER TABLE to add or drop columns, are renames, or have unique key added or dropped.

DEFAULTUSERPASSWORD is valid for Replicat. It specifies a different password for replicated {CREATE | ALTER} USER <name> IDENTIFIED BY <password> statement from the one used in the source statement. The password may be entered as a clear text or encrypted using the default or a user defined <keyname> from ENCKEYS. When using DEFAULTUSERPASSWORD, use the NOREPLICATEPASSWORD option of DDLOPTIONS for Extract.

GETAPPLOPS | IGNOREAPPLOPS are valid for Extract. This controls whether or not DDL operations produced by business applications *except* Replicat are included in the content that Extract writes to a trail or file. The default is GETAPPLOPS.

GETREPLICATES | IGNOREREPLICATES is valid for Extract. It controls whether or not DDL operations produced by Replicat are included in the content that Extract writes to a trail or file. The default is IGNOREREPLICATES.

REMOVECOMMENTS is valid for Extract and Replicat. It controls whether or not comments are removed from the DDL operation. By default, comments are not removed.

REMOVECOMMENTS BEFORE removes comments before the DDL operation is processed by Extract or Replicat. AFTER removes comments after they are used for string substitution.

REPLICATEPASSWORD is valid for Extract. It applies to the password in a {CREATE | ALTER} USER <user> IDENTIFIED BY <password> command. By default GoldenGate uses the source password in the target CREATE or ALTER statement. To prevent the source password from being sent to the target, use NOREPLICATEPASSWORD.

## 11.8 Error Handling

DDLERROR parameter: default and specific error handling rules to handle full range of anticipated errors

Extract syntax:

***DDLERROR***

***[, RESTARTSKIP <num skips>]***

Replicat syntax:

**DDLERROR**

**{<error> | DEFAULT} {<response>}**

**[RETRYOP MAXRETRIES <n> [RETRYDELAY <delay>]]**

**{INCLUDE <clause> | EXCLUDE <clause>}**

**[, IGNOREMISSINGTABLES | ABENDONMISSINGTABLES]**

**[, RESTARTCOLLISIONS | NORESTARTCOLLISIONS]**

Where <response> can be IGNORE, ABEND, DISCARD

**Example:**

***ddlerror <error code> discard***

***ddlerror default ignore include OBJTYPE TABLE OBJNAME "hr1.\*"***

# 12. Configuration Options

- BATCHSQL
- Compression
- Encryption
- Event actions
- Reporting

## 12.1 BATCHSQL

### 12.1.1 Overview

- Used with Replicat process
- Supported for Oracle, DB2 LUW, DB2 on z/OS, Teradata, SQL Server and Sybase
- Batches similar SQL statements into arrays, as opposed to individual operations
- Operations containing the same table, operation type (I, U, D), and column list are grouped into a batch
- Each statement type is prepared once, cached, and executed many times with different variables
- Referential integrity is preserved
- Can be used with change capture or initial loads

#### **Qualifying criteria for BATCHSQL**

Operations containing the same table, operation type (I, U, D), and column list are grouped into a batch.

#### **Parent child referential integrity**

GoldenGate analyzes parent-child foreign key referential dependencies in the batches before executing them. If referential dependencies exist for statements that are in different batches, more than one statement per batch may be required to maintain the referential integrity.

#### **Syntax**

## **BATCHSQL**

**[BATCHERRORMODE | NOBATCHERRORMODE]**

**[BATCHESPERQUEUE <n>]**

**[BATCHTRANSOPS <n>]**

**[BYTESPERQUEUE <n>]**

**[OPSPERBATCH <n>]**

**[OPSPERQUEUE <n>]**

**[TRACE]**

### **BATCHERRORMODE | NOBATCHERRORMODE**

Set the response of Replicat to errors.

- In NOBATCHERRORMODE (default), Replicat aborts the transaction on an error, temporarily disables BATCHSQL, and retries in normal mode.
- In BATCHERRORMODE, Replicat attempts to resolve errors without reverting to normal mode.
- Requires HANDLECOLLISIONS to prevent Replicat from exiting on an error.

### **BATCHESPERQUEUE <n>**

Sets a maximum number of batches per queue before flushing all batches. The default is 50. **Note:** A queue is a thread of memory containing captured operations waiting to be batched. By default, there is one buffer queue, but you can change this with NUMTHREADS.

### **BATCHTRANSOPS <n>**

Controls the size of a batch. Set to the default of 1000 or higher.

### **BYTESPERQUEUE <n>**

Sets the maximum number of bytes to hold in a queue before flushing batches. The default is 20 megabytes.

### **OPSPERBATCH <n>**

Sets the maximum number of rows that can be prepared for one batch before flushing. The default is 1200.

### **OPSPERQUEUE <n>**

Sets the maximum number of row operations that can be queued for all batches before flushing. The default is 1200.

### **TRACE**

Enables tracing of BATCHSQL activity to the console and report file.

## **Managing the buffer**

Buffering consumes memory. To attain the optimum balance between efficiency and the use of memory, use the following options to control when a buffer is flushed:

BATCHESPERQUEUE, BYTESPERQUEUE, OPSPERBATCH, OPSPERQUEUE, BATCHTRANSOPS (the last buffer flush for a target transaction occurs when the threshold set with BATCHTRANSOPS is reached).

### **Getting advantage out of BATCHSQL**

The row size should be smaller

- At 100 bytes of data per row change, BATCHSQL has been known to improve Replicat's performance from 400 to 500 percent.
- At around 5,000 bytes of data per row change, BATCHSQL benefits diminish.

More Similar operations with same table name, column list and operation type

### **Usage restrictions**

Some statement types cannot be processed in batches and must be processed as exceptions. When BATCHSQL encounters them, it flushes everything in the batch, applies the exceptions in the normal manner of one at a time, and then resumes batch processing. Transaction integrity is maintained. Statements treated as exceptions include:

- Statements containing LOB or LONG data.
- Statements containing rows longer than 25k in length.
- Statements where the target table has one or more unique keys besides the primary key.

## **12.2 Compression**

### **12.2.1 Overview**

- GoldenGate provides optional data compression when sending data over TCP/IP
- Automatic decompression is performed by Server Collector on remote system
- Compression threshold allows user to set minimum block size for which to compress
- GoldenGate uses the zlib compression.

### **Example:**

Compression is specified on the Extract RMTHOST parameter:

**RMTHOST newyork, MGRPORT 7809, COMPRESS, COMPRESSTHRESHOLD 750**

**COMPRESS** specifies that outgoing blocks of captured changes are compressed. **COMPRESSTHRESHOLD** sets the minimum byte size for which compression will occur. Default is 1000 bytes.

The destination Server Collector decompresses the data stream before writing it to the remote file or remote trail.

## 12.3 Encryption

- Message Encryption
  - Encrypts the messages sent over TCP/IP
  - Uses Blowfish, a symmetric 64-bit block cipher from CounterPane™ Internet Security
  - The data is automatically decrypted by Server Collector before saving the data to the trail
- Trail or Extract File Encryption
  - GoldenGate uses 256-key byte substitution
  - Encrypts only the record data in a trail or extract file
  - The data is decrypted by a downstream data pump or Replicat
- Database Password Encryption
  - Encrypted password can be generated using a default key or userdefined key



### 12.3.1 Message Encryption

1. Run the GoldenGate KEYGEN utility to generate random hex keys

**C:\GG> RUN KEYGEN <key length> <number of keys>**

Blowfish accepts a variable-length key from 32 to 128 bits

2. Enter key names and values in an ASCII text file named ENCKEYS (upper case, no file extension) in the GoldenGate install directory

**##Key name Key value**

**superkey 0x420E61BE7002D63560929CCA17A4E1FB**

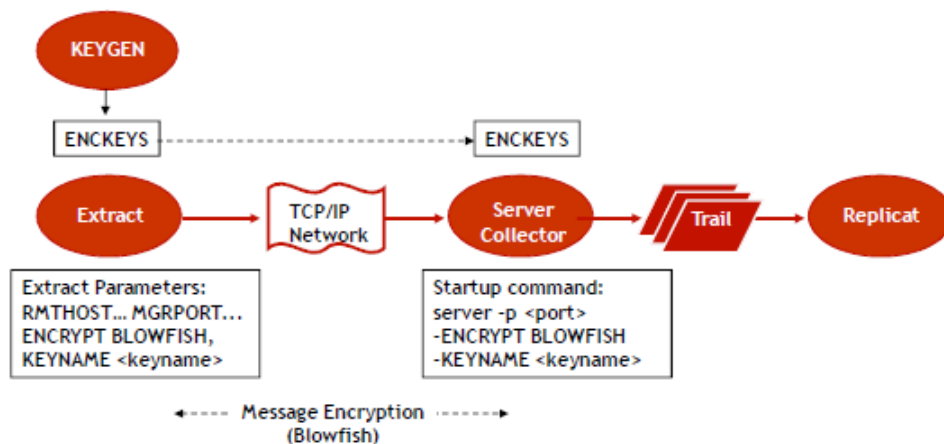
**secretkey 0x027742185BBF232D7C664A5E1A76B040**

3. Copy the ENCKEYS file to the source and target GoldenGate install directory
4. In the Extract parameter files, use the RMTHOST ENCRYPT and KEYNAME parameters

**RMTHOST West, MGRPORT 7809, ENCRYPT BLOWFISH, KEYNAME superkey**

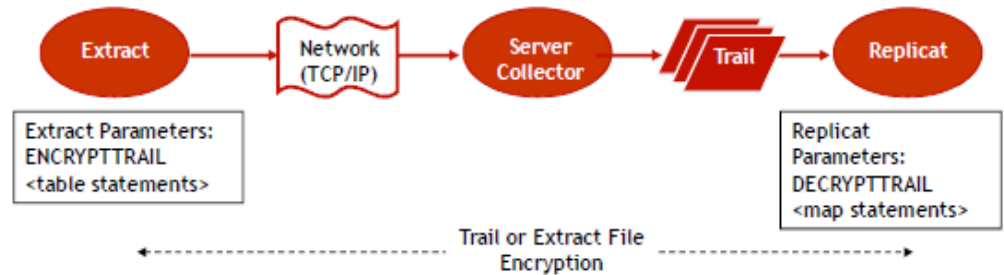
5. Configure a static Server Collector and start it manually with the -ENCRYPT and -KEYNAME parameters

**server -p <port> -ENCRYPT BLOWFISH -KEYNAME <keyname>**



### 12.3.2 Trail or Extract File Encryption

- Set Extract ENCRYPTTRAIL and Replicat DECRYPTTRAIL parameters
- Can set ENCRYPTTRAIL before tables you want encrypted and NOENCRYPTTRAIL before other tables
- Downstream data pumps can also decrypt the trail for transformation and pass it on either encrypted or decrypted



### 12.3.3 Password encryption

#### 12.3.3.1 Using Default Key

1. Generate an encrypted password with a GoldenGate default key code:

```
GGSCI> ENCRYPT PASSWORD <password>
```

For example:

```
GGSCI> ENCRYPT PASSWORD goldenpassword
```

No key specified, using default key...

Encrypted password: AACAAAAAAAAAAAAOARAQIDGEEEXAFAQJ

2. Paste the encrypted password in the Extract or Replicat PASSWORD parameter, for example:

```
SOURCEDB MySource, USERID joe, PASSWORD
```

```
AACAAAAAAAAAAAAOARAQIDGEEEXAFAQJ, ENCRYPTKEY DEFAULT
```

### 12.3.3.2 Using User defined Key

1. Generate an encrypted password with a user-defined key:

```
GGSCI> ENCRYPT PASSWORD <password>, ENCRYPTKEY <keyname>
```

For example:

```
GGSCI> ENCRYPT PASSWORD MyPass, ENCRYPTKEY DRKEY
```

**Encrypted password: AACAAAAAAAAAAAAIAJFGBNEYGTGSBSHVB**

2. Enter the key name and value in the ENCKEYS file, for example:

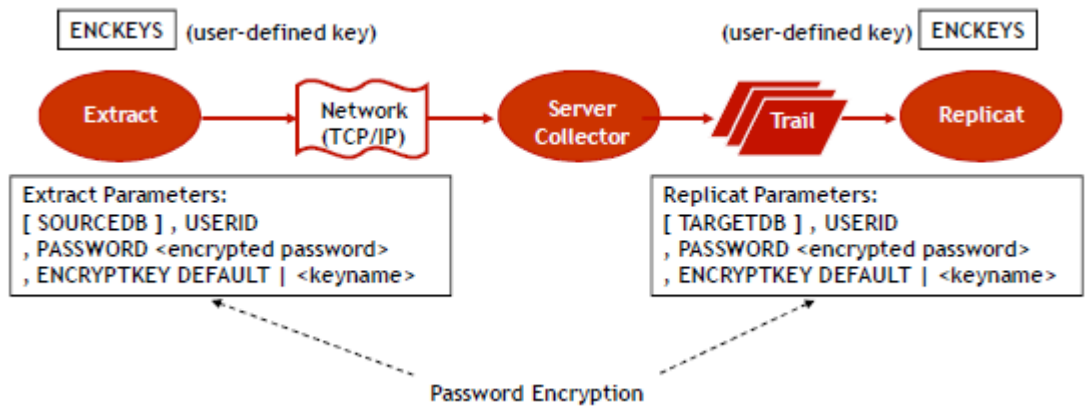
```
##Key name Key value
```

```
drkey 0x11DF0E2C2BC20EB335CB98F05471A737
```

3. Paste the encrypted password in the Extract or Replicat PASSWORD parameter, for example:

```
SOURCEDB MySource, USERID joe, PASSWORD
```

```
AACAAAAAAAAAAAAIAJFGBNEYGTGSBSHVB, ENCRYPTKEY drkey
```



## 12.4 Event Actions

### 12.4.1 Overview

- GoldenGate provides an event marker system that enables the GoldenGate processes to take a defined action based on an event record in the transaction log or trail
- The event record is:
  - Either a record in a data table that satisfies a filter condition for which you want an action to occur
  - Or a record that you write to a dedicated event table when you want an action to occur
- Only implemented for change replication, not initial loads
- EVENTACTION can specify one or multiple events

### 12.4.2 Actions you may take

- Stop the process
- Ignore or discard the current record
- Log an informational or warning message to the report file, GoldenGate error log and system event log
- Generate a report file
- Rollover the trail file
- Run a shell command – for example, to switch an application, start batch processes or start end-of-day reporting
- Activate tracing
- Write a checkpoint before and/or after writing the record to the trail

### 12.4.3 Examples

Example using a separate event table to manage events:

```
TABLE source.event_table, EVENTACTION (ROLLOVER);
```

Whenever a record is written to the event table, the trail file is rolled over.

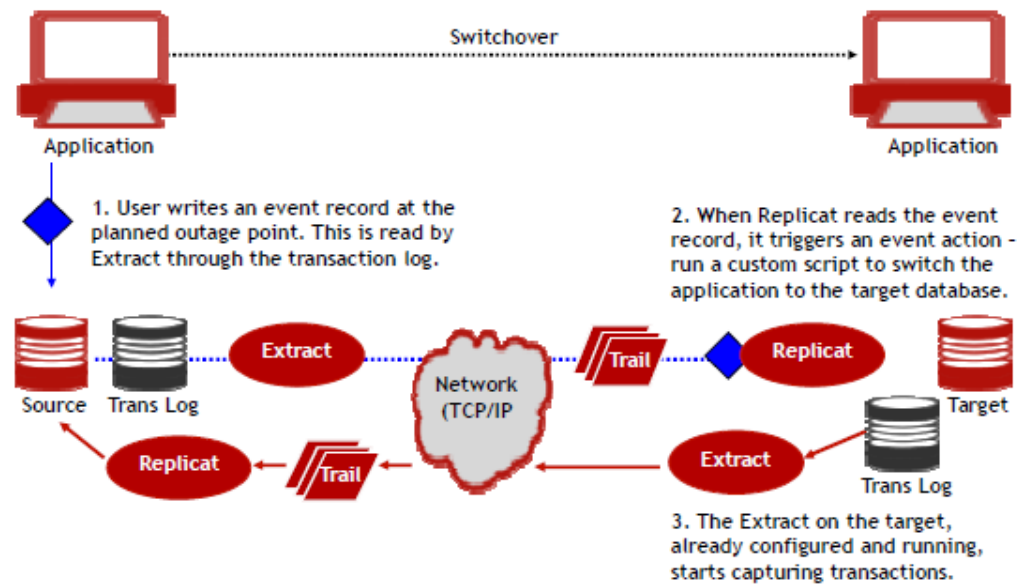
Example using data values to trigger events:

```
MAP source.account, TARGET target.account,
```

```
FILTER (account_no = 100), EVENTACTIONS (DISCARD, LOG);
```

Any record where account\_no = 100 is discarded and a log message written.

#### Automated Switchover Example



#### 12.4.4 Syntax

*TABLE | MAP*

...

*EVENTACTIONS (*

*[STOP | ABORT | FORCESTOP]*

*[IGNORE [TRANSACTION [INCLUDEVENT]]]*

*[DISCARD]*

*[LOG [INFO | WARNING]]*

*[REPORT]*

*[ROLLOVER]*

*[SHELL <command>]*

*[TRACE <trace file> [TRANSACTION] [PURGE | APPEND]]*

*[CHECKPOINT [BEFORE | AFTER | BOTH]]*

*[, ...]*

*)*

STOP – Graceful stop

ABORT – Immediate exit

FORCESTOP – Graceful stop if the event record is the last operation in the transaction, else log warning message and abort

IGNORE [TRANSACTION [INCLUDEVENT]] – Ignore record. Optionally ignore entire transaction and propagate the event record.

DISCARD – Write record to discard file

LOG [INFO | WARNING] – Log an informational or warning message to the report, error and systems event files

REPORT – Generate a report file

ROLLOVER – (Extract only) Roll over the trail file

SHELL – Execute a shell command

TRACE – Write trace information to file

CHECKPOINT – Write a checkpoint before and/or after writing the event record

## 12.5 Reporting

- Set up hourly or daily interval reports via parameters
  - **REPORT**
  - **REPORTCOUNT**
  - **REPORTROLLOVER**
- Generate reports on demand via commands
  - **SEND [ EXTRACT | REPLICAT ] <group>, REPORT**
- View process reports and history of reports
  - **VIEW REPORT <group>**
  - **VIEW REPORT <group>[n]**
  - **VIEW REPORT <filename>**

### Statistics – Overview

- Generate statistics on demand
  - **STATS [ EXTRACT | REPLICAT ] {group}**
- View latest statistics
  - **STATS <group>, LATEST**
- Display daily statistics
  - **STATS <group>, DAILY**
- Other statistics options
  - By table
  - Totals only
  - Hourly
  - Reset statistics
  - Report rate

- Report fetch statistics (Extract)
- Report collisions (Replicat)
- STATOPTIONS parameter
- (report fetch statistics or collisions, reset statistics on report rollover)

### **STATS Command**

- STATS [ EXTRACT | REPLICAT ] {group} or STATS {group}
- STATS <group>, <statistic>
  - TOTAL
  - DAILY
  - HOURLY
  - LATEST
  - RESET
- STATS <group>, <option>
  - TABLE {owner.tablename}
  - TOTALONLY
  - REPORTRATE
  - REPORTFETCH | NOREPORTFETCH (Extract only)
  - REPORTDETAIL | NOREPORTDETAIL (Replicat only)



## Zero Downtime Upgrade Steps

- ▶ Configure and Start Extract and data pump
- ▶ **Take backup of source**
- ▶ **Transport backup to target**
- ▶ **Restore the backup on target**
- ▶ **Recover your database using the archivelogs and open it.**
- ▶ Start replicat resolve any errors if their (use of HANDLECOLLISIONS parameter)
- ▶ Test your replica
- ▶ Move your application to point to the replica