Oracle® Fusion Middleware

Installing and Configuring Oracle GoldenGate for Oracle Database

12*c* (12.2.0.1)
E66365-07
July 2018

ORACLE

Oracle Fusion Middleware Installing and Configuring Oracle GoldenGate for Oracle Database, 12*c* (12.2.0.1)

E66365-07

Contents

9 Configuring Oracle GoldenGate Apply

10 Additional Oracle GoldenGate Configuration Considerations

11 Additional Configuration Steps for Using Classic Capture

Preface

This guide helps you get started with installing, configuring, and running Oracle GoldenGate on an Oracle Database system. With Oracle GoldenGate for Oracle, you can:

- Map, filter, and transform transactional data changes between similar or dissimilar supported Oracle versions, and between supported Oracle versions and other supported types of databases.
- Replicate and filter Oracle DDL operations between heterogeneous Oracle Databases.
- Perform initial loads to target tables in Oracle or other databases to instantiate a synchronized replication environment.

This documentation is meant to be a step by step guide in establishing a basic Oracle GoldenGate configuration, from source to target, that is tailored to the Oracle environment. It should be followed in a linear fashion, as appropriate for the capture and apply methods that you select, so that you benefit from important information in previous sections. It is not meant to be used as a reference document. Where needed, it points you to other documentation where you can find additional information to expand the configuration to suit your needs.

Audience

This guide is intended for installers, database administrators, and system administrators who are installing, configuring and running Oracle GoldenGate.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Accessible Access to Oracle Support

Oracle customers who have purchased support have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Information

The Oracle GoldenGate Product Documentation Libraries are found at

Oracle GoldenGate

Oracle GoldenGate Application Adapters

Oracle GoldenGate for Big Data

Oracle GoldenGate Plug-in for EMCC

Oracle GoldenGate Monitor

Oracle GoldenGate for HP NonStop (Guardian)

Oracle GoldenGate Veridata

Oracle GoldenGate Studio

Additional Oracle GoldenGate information, including best practices, articles, and solutions, is found at:

Oracle GoldenGate A-Team Chronicles

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, such as "From the File menu, select **Save**." Boldface also is used for terms defined in text or in the glossary.
italic *italic*	Italic type indicates placeholder variables for which you supply particular values, such as in the parameter statement: TABLE *table_name*. Italic type also is used for book titles and emphasis.
monospace MONOSPACE	Monospace type indicates code components such as user exits and scripts; the names of files and database objects; URL paths; and input and output text that appears on the screen. Uppercase monospace type is generally used to represent the names of Oracle GoldenGate parameters, commands, and user-configurable functions, as well as SQL commands and keywords.
UPPERCASE	Uppercase in the regular text font indicates the name of a utility unless the name is intended to be a specific case.
{ }	Braces within syntax enclose a set of options that are separated by pipe symbols, one of which must be selected, for example: {*option1* | *option2* | *option3*}.
[ ]	Brackets within syntax indicate an optional element. For example in this syntax, the SAVE clause is optional: CLEANUP REPLICAT *group_name* [, SAVE *count*]. Multiple options within an optional element are separated by a pipe symbol, for example: [*option1* | *option2*].

1

System Requirements and Preinstallation Instructions

This chapter contains the requirements for the system and database resources that support Oracle GoldenGate.

Topics:

1.1 Verifying Certification and System Requirements

Make sure that you are installing your product on a supported hardware or software configuration. For more information, see the certification document for your release on the *Oracle Fusion Middleware Supported System Configurations* page.

Oracle has tested and verified the performance of your product on all certified systems and environments; whenever new certifications occur, they are added to the proper certification document right away. New certifications can occur at any time, and for this reason the certification documents are kept outside of the documentation libraries and are available on Oracle Technology Network.

1.2 Operating System Requirements

This section outlines the operating system resources that are necessary to support Oracle GoldenGate.

- Other Programs (page 1-6)

1.2.1 Memory Requirements

The amount of memory that is required for Oracle GoldenGate depends on the amount of data being processed, the number of Oracle GoldenGate processes running, the amount of RAM available to Oracle GoldenGate, and the amount of disk space that is available to Oracle GoldenGate for storing pages of RAM temporarily on disk when the operating system needs to free up RAM (typically when a low watermark is reached). This temporary storage of RAM to disk is commonly known as *swapping* or *paging* (herein referred to as *swapping*). Depending on the platform, the term *swap space* can be a swap partition, a swap file, a page file (Windows) or a shared memory segment (IBM i platforms).

Modern servers have sufficient RAM combined with sufficient swap space and memory management systems to run Oracle GoldenGate. However, increasing the amount of RAM available to Oracle GoldenGate may significantly improve its performance, as well as that of the system in general.

Typical Oracle GoldenGate installations provide RAM in multiples of gigabytes to prevent excessive swapping of RAM pages to disk. The more contention there is for RAM the more swap space that is used.

Excessive swapping to disk causes performance issues for the Extract process in particular, because it must store data from each open transaction until a commit record is received. If Oracle GoldenGate runs on the same system as the database, the amount of RAM that is available becomes critical to the performance of both.

RAM and swap usage are controlled by the operating system, not the Oracle GoldenGate processes. The Oracle GoldenGate cache manager takes advantage of the memory management functions of the operating system to ensure that the Oracle GoldenGate processes work in a sustained and efficient manner. In most cases, users need not change the default Oracle GoldenGate memory management configuration.

For more information about evaluating Oracle GoldenGate memory requirements, see the `CACHEMGR` parameter in *Reference for Oracle GoldenGate for Windows and UNIX*.

1.2.2 Disk Requirements

Assign an amount of free disk space based on the platform, database, and whether you are installing Oracle GoldenGate in the normal manner or with the Oracle Universal Installer (OUI). You can delete the download file after the installation is complete.

- Disk Requirements for Oracle GoldenGate Installation Files (page 1-2)
- Other Disk Space Considerations (page 1-3)
- Installing in a Cluster (page 1-4)
- Temporary Disk Requirements (page 1-4)

1.2.2.1 Disk Requirements for Oracle GoldenGate Installation Files

This section shows the disk requirements for a manual installation and for an installation through OUI.

Table 1-1 (page 1-3) shows the disk space that is consumed by the files of one Oracle GoldenGate installation in a manual build. A manual build does not use OUI. The person installing Oracle GoldenGate uncompresses the files and creates the working directories.

Table 1-1 Disk Requirements for a Manual Build

Platform	Oracle Version	Compressed Size (MB)	Installed Size (MB)
Windows	11g	92.3 zip file	188
Windows	12c	92.3 zip file	188
Linux	11g	137 zip file 319 tar file	319
Linux	12c	146 zip file 348 tar file	348
Solaris SPARC	11g	62 zip file 194 tar file	199
Solaris SPARC	12c	87 zip file 198 tar file	203

Table 1-2 (page 1-3) shows the disk space that is consumed by the files of one Oracle GoldenGate installation when OUI is used.

Table 1-2 Disk Requirements for an OUI Build

Platform	Oracle Version	Compressed Size (MB)	Installed Size (MB)
Windows	11g	321 zip file 325 OUI installer	504
Windows	12c	321 zip file 325 OUI installer	504
Linux	11g	325 zip file 329 OUI installer	492
Linux	12c	325 zip file 329 OUI installer	521
Solaris SPARC	11g	214 zip file 219 OUI installer	391
Solaris SPARC	12c	214 zip file 219 OUI installer	391

1.2.2.2 Other Disk Space Considerations

In addition to the disk space required for the files and binaries that are installed by Oracle GoldenGate, allow an additional 1 GB of disk space on any system that hosts the Oracle GoldenGate trail (or trails). A trail is a set of self-aging files that contain the working data at rest and during processing. You may need more or less than this amount, because the space that is consumed by the trails depends on the volume of data that will be processed. See the guidelines for sizing trails in *Administering Oracle GoldenGate for Windows and UNIX*.

Disk space is also required for the Oracle GoldenGate Bounded Recovery feature. Bounded Recovery is a component of the general Extract checkpointing facility. It caches long-running open transactions to disk at specific intervals to enable fast recovery upon a restart of Extract. At each bounded recovery interval (controlled by the `BRINTERVAL` option of the `BR` parameter) the disk required is as follows: for each transaction with cached data, the disk space required is usually 64k plus the size of the cached data rounded up to 64k. Not every long-running transaction is persisted to disk. For complete information about Bounded Recovery, see the `BR` parameter in *Reference for Oracle GoldenGate for Windows and UNIX*.

1.2.2.3 Installing in a Cluster

To install Oracle GoldenGate into a cluster environment using the Oracle GoldenGate MASA, you need only to install the Oracle GoldenGate read only home directory and then create custom deployment specific directories. For more information, see Preparing to Install Oracle GoldenGate Within a Cluster.

1.2.2.4 Temporary Disk Requirements

By default, Oracle GoldenGate maintains data that it writes to disk in the `dirtmp` sub-directory of the Oracle GoldenGate installation directory. When total cached transaction data exceeds the `CACHESIZE` setting of the `CACHEMGR` parameter, Extract will begin writing cache data to temporary files. The cache manager assumes that all of the free space on the file system is available. This directory can fill up quickly if there is a large transaction volume with large transaction sizes. To prevent I/O contention and possible disk-related Extract failures, dedicate a disk to this directory. You can assign a name to this directory with the `CACHEDIRECTORY` option of the `CACHEMGR` parameter.

It is typically more efficient for the operating system to swap to disk than it is for Extract to write temporary files. The default `CACHESIZE` setting assumes this. Thus, there should be sufficient disk space to account for this, because only after the value for `CACHESIZE` is exceeded will Extract write transaction cached data to temporary files in the file system name space. If multiple Extract processes are running on a system, the disk requirements can multiply. Oracle GoldenGate writes to disk when there is not enough memory to store an open transaction. Once the transaction has been committed or rolled back, committed data is written to trail files and the data are released from memory and Oracle GoldenGate no longer keeps track of that transaction. There are no minimum disk requirements because when transactions are committed after every single operations these transactions are never written to disk.

1.2.3 Network

The following network resources must be available to support Oracle GoldenGate.

- For optimal performance and reliability, especially in maintaining low latency on the target, use the fastest network possible and install redundancies at all points of failure.
- Configure the system to use both TCP/IP and UDP services, including DNS. Oracle GoldenGate supports IPv4 and IPv6 and can operate in a system that supports one or both of these protocols.
- Configure the network with the host names or IP addresses of all systems that will be hosting Oracle GoldenGate processes and to which Oracle GoldenGate will be connecting. Host names are easier to use.

- Oracle GoldenGate requires some unreserved and unrestricted TCP/IP ports, the number of which depends on the number and types of processes in your configuration. See *Administering Oracle GoldenGate for Windows and UNIX* for details on how to configure the Manager process to handle the required ports.
- Keep a record of the ports that you assigned to Oracle GoldenGate. You will specify them with parameters when configuring the `oggca.sh` process similar to configuring the Manager process in the classic architecture.
- Configure your firewalls to accept connections through the Oracle GoldenGate ports.

1.2.4 Operating System Privileges

The following are the privileges in the operating system that are required to install Oracle GoldenGate and to run the processes.

- To install on Windows, the person who installs Oracle GoldenGate must log in as Administrator.
- To install on UNIX, the person who installs Oracle GoldenGate must have read and write privileges on the Oracle GoldenGate installation directory.
- The Oracle GoldenGate Extract, Replicat, and Manager processes must operate as an operating system user that has privileges to read, write, and delete files and subdirectories in the Oracle GoldenGate directory. In addition, the Manager process requires privileges to control the other Oracle GoldenGate processes.
- (Classic capture mode) In classic capture mode, the Extract process reads the redo logs directly and must operate as an operating system user that has read access to the log files, both online and archived. On UNIX systems, that user must be a member of the group that owns the Oracle instance. If you install the Manager process as a Windows service during the installation steps in this documentation, you must install as Administrator for the correct permissions to be assigned. If you cannot install Manager as a service, assign read access to the Extract process manually, and then always run Manager and Extract as Administrator.
- Dedicate the Extract, Replicat, and Manager operating system users to Oracle GoldenGate. Sensitive information might be available to anyone who runs an Oracle GoldenGate process, depending on how database authentication is configured.

1.2.5 Itanium Requirements

To install Oracle GoldenGate on an Intel Itanium system, the `vcredist_IA64.exe` runtime library package must be installed. You can download this package from the Microsoft website. This package includes Visual Studio DLLs necessary for Oracle GoldenGate to operate on the Itanium platform. If these libraries are not installed, Oracle GoldenGate generates the following error.

```
The application failed to initialize properly (0xc0150002). Click on Ok to terminate
the application.
```

1.2.6 Console Character Sets

The operating system and the command console must have the same character sets. Mismatches occur on Microsoft Windows systems, where the operating system is set

to one character set, but the DOS command prompt uses a different, older DOS character set. Oracle GoldenGate uses the character set of the operating system to send information to GGSCI command output; therefore a non-matching console character set causes characters not to display correctly. You can set the character set of the console before opening a GGSCI session by using the following DOS command:

```
chcp OS character set
```

If the characters do not display correctly after setting the code page, try changing the console font to Lucida Console, which has an extended character set.

1.2.7 Other Programs

The following are additional considerations in support of Oracle GoldenGate.

- Before installing Oracle GoldenGate on a Windows system, install and configure the Microsoft Visual C ++ 2010 SP1 Redistributable Package. **Make certain it is the SP1 version of this package, and make certain to get the correct bit version for your server.** This package installs runtime components of Visual C++ Libraries. For more information and to download this package, go to `http://www.microsoft.com`.
- Oracle has not certified any of its products on VMware virtualized environments. Oracle Support will assist customers running Oracle products on VMware in the following manner: Oracle will only provide support for issues that either are known to occur on the native OS, or can be demonstrated not to be as a result of running on VMware.

1.3 Oracle Universal Installer Requirements for Oracle GoldenGate

To use Oracle Universal Installer to install Oracle GoldenGate, the following requirements must be satisfied before running the installer.

- OUI is a graphical Java application. The host platform must be Java-enabled. The required Java Runtime Environment (JRE) is automatically installed with OUI.
- If this is the first time you are installing an Oracle product with OUI, you will be prompted to create the Oracle central inventory, which also creates a subdirectory for Oracle GoldenGate to contain inventory information. This subdirectory typically requires 150 kilobytes of disk space.

1.4 Database Configuration

This section contains Oracle GoldenGate requirements that are specific to the Oracle Database. These apply to both capture modes unless explicitly noted.

- If you are using the `DBMS_LOB.LOADFROMFILE` procedure to update a LOB column only and your supplemental log is on all the columns, Integrated Extract captures the key columns and LOB improving performance. Classic Extract captures all the columns by default. These behaviors do not affect like to like replications. However, with a replication to data warehouse, all the columns might have to be

updated. If you are converting from Classic Extract to Integrated Extract, you must use one of the following parameters to ensure that the Extract operates correctly:

  - Use `KEYCOLS` to add all columns (except LOB).
  - Use `LOGALLSUPCOLS` to control the writing of supplementally logged columns.

- Database user privileges and configuration requirements are explained in Establishing Oracle GoldenGate Credentials in *Installing and Configuring Oracle GoldenGate for Oracle Database*.
- If the database is configured to use a bequeath connection, the `sqlnet.ora` file must contain the `bequeath_detach=true` setting.
- Oracle Databases must be in `ARCHIVELOG` mode so that Extract can process the log files.
- Ensure that your database has `SUPPLEMENTAL LOGGING` enabled.

1.5 Summary of Supported Oracle Data Types and Objects Per Capture Mode

Table 1-3 (page 1-7) summarizes the way that Oracle GoldenGate supports the Oracle data types according to the capture mode that you choose. For more information about capture modes, see Deciding Which Capture Method to Use. (page 5-2).

Detailed support information for Oracle data types, objects, and operations starts with Details of Support for Oracle Data Types (page 1-11).

Table 1-3 Supported Data Types Per Capture Mode

Data type	Classic capture	Integrated capture
Scalar columns including `DATE` and `DATETIME` columns	Captured from redo.	Captured from redo.
`LONG VARCHAR`	Not supported.	Captured from redo.
`BASICFILE` LOB columns	LOB modifications done using DML (`INSERT`/`UPDATE`/`DELETE`) are captured from redo. LOB modifications done using `DBMS_LOB` package are captured from the source table by fetching values from the base table.	Captured from redo.

Table 1-3 (Cont.) Supported Data Types Per Capture Mode

Data type	Classic capture	Integrated capture
SECUREFILE LOB columns	Captured from redo, except for the following cases where SECUREFILE LOBs are fetched from the source table: • LOB is encrypted • LOB is compressed • LOB is deduplicated • LOB is stored in-line • LOB is modified using DBMS_LOB package • NOLOGGING LOBs	Captured from redo, except for the following cases where SECUREFILE LOBs are fetched from the source table: • LOBs is modified using DBMS_LOB.FRAGMENT_* procedures. • NOLOGGING LOBs. • Deduplicated LOBs when the source Oracle Database release/binary is less than 12.1. Requires source database compatibility to be set to 11.2.0.0.0 or higher
Index Organized Tables (IOT)	Captured from redo with the following restrictions: • IOT with mapping table not supported. • Direct load inserts to IOT tables cannot have the SORTED clause. • IOT with prefix compression as specified with COMPRESS clause is not supported.	Captured from redo with the following restriction: • IOT with mapping table not supported.
XML columns stored as CLOB	Captured from redo.	Captured from redo. Requires source database compatibility to be set to 11.0.0.0.0 or higher
XML columns stored as Binary	Fetched from source table.	Requires source database compatibility to be set to 11.2.0.3.0 or higher. Fetched from source table if compatibility is less than 11.2.0.3.0.
XML columns stored as Object-Relational	Not supported.	Captured from redo. Requires source database compatibility to be set to 11.2.0.3.0 or higher.
XML Type Table	Not supported.	Captured from redo.

Table 1-3 (Cont.) Supported Data Types Per Capture Mode

Data type	Classic capture	Integrated capture
User Defined Type (UDT) columns	Fetched from source table.	Captured from redo with limitations. Specify `TRANLOGOPTIONS USENATIVEOBJSUPPORT` to enable the capture from redo. Use of Native Object Support requires source database compatibility to be set to 12.0.0.0.0 or higher. Fetched from source table when: • `USENATIVEOBJSUPPORT` is not specified • If the redo compatibility is less than 12.0.0.0.0 • If the `UDT` contains Nested Table, `SDO_TOPO_GEOMETRY`, or `SDO_GEORASTER` types Procedural supplemental logging must be enabled at the source so that TOPO and Georaster can be supported.
Invisible Columns	Not supported.	

`ANYDATA` columns	Fetched from source table with the following data types only:
	`BINARY_DOUBLE`
	`BINARY_FLOAT`
	`CHAR`
	`DATE`
	`INTERVAL DAY TO SECOND`
	`INTERVAL YEAR TO MONTH`
	`NCHAR`
	`NUMBER`
	`NVARCHAR2`
	`RAW`
	`TIMESTAMP`
	`TIMESTAMP WITH TIME ZONE`
	`TIMESTAMP WITH LOCAL TIMEZONE`
	UDTs
	`VARCHAR/VARCHAR2`
	Requires source database compatibility to be set to 11.2.0.0.0 or higher.
		Captured from redo with limitations.
		Specify `TRANLOGOPTIONS USENATIVEOBJSUPPORT` to enable the capture from redo.
		Use of Native Object Support requires source database compatibility to be set to 12.0.0.0.0 or higher.
		Fetched from source table when
		• "`USENATIVEOBJSUPPORT` is not specified
		• If the redo compatibility is less than 12.0.0.0.0

Table 1-3 (Cont.) Supported Data Types Per Capture Mode

Data type	Classic capture	Integrated capture
Spatial Types columns	Fetched from source table.	Captured from redo with limitations. Specify `TRANLOGOPTIONS USENATIVEOBJSUPPORT` to enable the capture from redo. Use of Native Object Support requires source database compatibility to be set to 12.2.0.1.0 or higher. Fetched from source table when: • `USENATIVEOBJSUPPORT` is not specified • If the redo compatibility is less than 12.0.0.0.0 • If `SDO_TOPO_GEOMETRY` or `SDO_GEORASTER` (raster tables) are used Procedural supplemental logging must be enabled at the source so that TOPO and Georaster can be supported.
Collections columns (`VARRAYs`)	Fetched from source table.	Captured from redo for `VARRAY` attributes `VARRAY` attributes of UDT types are supported. Specify `TRANLOGOPTIONS USENATIVEOBJSUPPORT` to enable the capture from redo. Requires source database compatibility to be set to 12.0.0.0.0 or higher. Fetched from source table when • `USENATIVEOBJSUPPORT` is not specified • the redo compatibility is less than 12.0.0.0.0 • If Top-level `VARRAY` columns are used, and compatibility is less than 12.2.
Collections columns (Nested Tables)	Fetched from source table with limitations. See Details of Support for Objects and Operations in Oracle DML (page 1-21).	Fetched from source table with limitations. See Details of Support for Objects and Operations in Oracle DML (page 1-21).

Table 1-3 (Cont.) Supported Data Types Per Capture Mode

Data type	Classic capture	Integrated capture
Object Table	Fetched from source table.	Captured from redo with limitations. Specify `TRANLOGOPTIONS USENATIVEOBJSUPPORT` to enable the capture from redo. Use of Native Object Support requires source database compatibility to be set to 12.0.0.0.0 or higher. Replication of DDL operations on an object table is not supported. Fetched from source table with additional limitations when • `USENATIVEOBJSUPPORT` is not specified • If the redo compatibility is less than 12.0.0.0.0 • If Nested Table, `SDO_TOPO_GEOMETRY` or `SDO_GEORASTER` (raster tables) are used
Transparent Data Encryption (Column Encryption & Tablespace Encryption)	Captured from redo.	Captured from redo. No additional setup is required for local capture. Requires source database compatibility to be set to 11.0.0.0.0 or higher.
Basic Compression	Not supported.	Captured from redo.
OLTP-Compression	Not supported.	Captured from redo.
Exadata Hybrid Columnar Compression	Not supported.	Captured from redo.
XA on non-RAC database	Captured from redo.	Captured from redo.
XA on RAC database	Not supported. To get support, must make sure all branches of XA goes to the same instance.	Captured from redo. Requires source database compatibility to be set to 11.2.0.0.0 or higher.
PDML on non-RAC database	Captured from redo.	Captured from redo.
PDML on RAC database	Not supported. To get support, you must make sure child transactions spawned from a PDML transaction do not span multiple instances.	Captured from redo.

1.6 Details of Support for Oracle Data Types

The following outlines details of Oracle data type support by Oracle GoldenGate. Unless otherwise noted, the support applies to both classic and integrated capture mode.

- ANYDATA Data Types (page 1-12)

1.6.1 ANYDATA Data Types

The following `ANYDATA` data types are supported:

Fetched from source table with the following data types only:

- `BINARY_DOUBLE`
- `BINARY_FLOAT`
- `CHAR`
- `DATEINTERVAL DAY TO SECOND`
- `INTERVAL YEAR TO MONTH`
- `NCHAR`
- `NUMBER`
- `NVARCHAR2`
- `RAW`
- `TIMESTAMP`
- `TIMESTAMP WITH TIME ZONE`
- `TIMESTAMP WITH LOCAL TIMEZONE`
- UDTs
- `VARCHAR/VARCHAR2`
- Limitations of Support (page 1-12)

1.6.1.1 Limitations of Support

- Your source database compatibility must be set to 11.2.0.0.0 or higher. Support for named collections and `VARRAYS` embedded within those data types.

1.6.2 Numeric Data Types

The following numeric data types are supported:

- `NUMBER` up to the maximum size permitted by Oracle
- `BINARY FLOAT`

- BINARY DOUBLE
- UROWID

  Classic Extract and Integrated Extract do not support UROWID.

- Limitations of Support (page 1-13)

1.6.2.1 Limitations of Support

The support of the range and precision for floating-point numbers depends on the host machine. In general, the precision is accurate to 16 significant digits, but you should review the database documentation to determine the expected approximations. Oracle GoldenGate rounds or truncates values that exceed the supported precision.

1.6.3 Character Data Types

The following character data types are supported:

- CHAR
- VARCHAR2
- LONG
- NCHAR
- NVARCHAR2
- Limitations of Support (page 1-13)

1.6.3.1 Limitations of Support

- If an extended VARCHAR column is part of unique index or constraint, then direct path inserts to this table may cause Replicat to abend with a warning. Verify that the extended VARCHAR caused the abend by checking all_indexes/all_ind_columns for a unique index or all_cons_columns/all_constraints for a unique constraint. Once you determine that an extended VARCHAR, you can temporarily drop the index or disable the constraint:

  For Unique Index:

  ```
  drop index t2u;
  ```

  For Unique Constraint:

  ```
  alter table v32ind modify constraint sys_c0010125 disable;
  ```

- Extended (32K) VARCHAR2 and NVARCHAR2 columns are supported when Extract is in integrated capture mode. All modes of Replicat support 32K VARCHAR2 and NVARCHAR2 columns. The following limitations apply:
  - Oracle GoldenGate does not support 32K VARCHAR2 and NVARCHAR2 columns as part of a key or unique index, nor as a column in a KEYCOLS clause of the TABLE or MAP parameter. 32K columns cannot be used as row identifiers because they are not supplementally logged even when part of a primary key.
  - 32K columns are not supported as resolution columns in a CDR (conflict resolution and detection) configuration nor as the basis for any other work that requires a column value to be present in the transaction log.

  - Oracle GoldenGate does not limit the number of 32K columns, but each trail record has a length limit of 4MB for inline records. The number of 32K columns that reaches this limit is approximately 160 columns, but the number of columns also depends on the actual size of the extended `VARCHAR2` column.

1.6.4 Multi-byte Character Types

The following multi-byte character types are supported:

- `NCHAR` and `NVARCHAR2` multi-byte character data types
- Multi-byte data stored in `CHAR` and `VARCHAR2` columns
- Limitations of Support (page 1-14)

1.6.4.1 Limitations of Support

- For Oracle GoldenGate to support multi-byte character data, the source and target databases must be logically identical in terms of schema definition for the tables and sequences being replicated. Transformation, filtering, and other manipulation are not supported. The character sets between the two databases must be one of the following:
  - Identical, for example SHIFT-JIS on the source and on the target.
  - Equivalent, which is not the same character set but containing the same set of characters, for example SHIFT-JIS and EUC-JP.
  - Target is superset of the source: For example, UNICODE is a superset of all character types, and therefore of any other character set.
- Multi-byte data is supported whether the length semantics are in bytes or characters.

For additional configuration requirements, see Handling Special Data Types (page 10-3).

1.6.5 Binary Data Types

The following binary data types are supported:

- `RAW`
- `LONG RAW`

1.6.6 Date and Timestamp Data Types

The following date and time data types are supported:

- `DATE`
- `TIMESTAMP` (see Limitations of support)
- Limitations of Support (page 1-14)

1.6.6.1 Limitations of Support

- Oracle GoldenGate does not support negative dates.

- `INTERVAL DAY` and `INTERVAL YEAR` are supported only if the size of the target column is equal to, or greater than, that of the source.
- Oracle GoldenGate supports the capture and replication of `TIMESTAMP WITH TIME ZONE` as a UTC offset (`TIMESTAMP '2011-01-01 8:00:00 -8:00'`).
- `TIMESTAMP WITH TIME ZONE` as `TZR` (Region ID) is supported for the replication of data changes, but not for initial loads, for `SQLEXEC`, or for operations where the column must be fetched from the database. In these cases, the region ID is converted to a time offset by the database when the column is selected. Replicat replicates the timestamp as date and time data with a time offset value.
- Oracle GoldenGate supports timestamp data from `0001/01/03 00:00:00` to `9999/12/31 23:59:59`. If a timestamp is converted from GMT to local time, these limits also apply to the resulting timestamp. Depending on the time zone, conversion may add or subtract hours, which can cause the timestamp to exceed the lower or upper supported limit.
- Oracle GoldenGate supports time offset values between +12:00 and -12:00.

To support `TIMESTAMP WITH TIME ZONE` specified as `TZR` properly, and also to handle `TIMESTAMP WITH LOCAL TIMEZONE` properly.

1.6.7 Large Object Data Types

The following large object types are supported:

- `CLOB`
- `NCLOB`
- `BLOB`
- `SECUREFILE` and `BASICFILE`
- General Limitations of Support — Integrated and Classic Capture Modes (page 1-15)
- Limitations of support — Classic Capture Mode (page 1-15)

1.6.7.1 General Limitations of Support — Integrated and Classic Capture Modes

- Oracle GoldenGate does not support the filtering, column mapping, or manipulation of large objects larger than 4K.
- Assuming DDL support is enabled, Oracle GoldenGate replicates the CTAS statement and allows it to select the data from the underlying target table(s). This is controlled using the `GET_CTAS_DMLS` parameter and applies to all typed tables. If the parameter is used, the OIDs are preserved.

1.6.7.2 Limitations of support — Classic Capture Mode

- `BASICFILE` option `LOB`s are captured from the redo log, but are fetched from the database in the following circumstances:
  - Extract determines the `LOB` is invalid.
  - The LOB data is not in the redo log, which occurs when the `BASICFILE LOB` is created with the `NOLOGGING` option.

  - The LOB is created with the CACHE option.
  - The LOB is only partially updated. Oracle GoldenGate does not support partial column data. Extract assumes LOB data to be incomplete if the LOB does not start with a LOB reset record, or if the LOB does not start at the first byte and does not end at the last byte, according to the new LOB length. Partial updates can be generated by the following OCI calls: OCILOBWrite(), OCILobAppend(), OCiLobCopy(), OCILobLoadFromFile(), OCILobTrim(), and by updates made through procedures in the dbms_lob package.
  - Extract detects an anomaly in the LOB data.
- SECUREFILE option LOBs are captured from the redo logs only when the update is complete and the LOB is not transformed (the column is not compressed or encrypted or deduplicated) and stored out-of-row. SECUREFILE LOBs are fetched from the database in the following circumstances:
  - The LOB is stored in-row.
  - The LOB is transformed either with compression or encryption.
  - The LOB is created with the CACHE attribute.
  - Extract determines that a LOB instance is invalid.
  - LOB data is missing from the redo log. This can occur if the LOB is created with any of following options: DEDUPLICATE, NOLOGGING, FILESYSTEM_LIKE_LOGGNG.
  - The LOB is updated using OCILOBWrite(), OCILobAppend(), OCiLobCopy(), OCILobLoadFromFile(), OCILobTrim(), or through procedures in the dbms_lob package.
  - Any other anomalies as detected by Extract.
- When changing a SECUREFILE LOB from one storage to another (such as from ENCRYPT to DECRYPT), Oracle updates the whole table, and Extract captures those updates from the log. Therefore, it will appear as though Oracle updated all of the data blocks that are associated with the table. This also can happen when an ALTER TABLE command sets a DEFAULT value to a column that has null values.
- In a manage bundled agents (XAG) high availability environment that has tables containing JavaScript Object Notation (JSON) columns, Extracts can extract this data though the default Replicat mode cannot replicate the data. You must set the DBOPTIONS NOSKIPTEMPLOB parameter to avoid Replicat abending.

1.6.8 XML Data Types

The following XML types are supported:

- In integrated capture mode, Oracle GoldenGate supports XMLType columns and XMLType tables stored as XML CLOB, XML Object Relational, and XML Binary.
- In classic capture mode, Oracle GoldenGate supports XMLType columns stored as XML CLOB and XML Binary.

- Limitations of Support — Integrated and Classic Capture Modes (page 1-17)
- Limitations of Support — Integrated Capture Mode (page 1-17)
- Limitations of Support — Classic Capture Mode (page 1-17)

1.6.8.1 Limitations of Support — Integrated and Classic Capture Modes

The following are not supported:

- Filtering and manipulation are not supported. You can map the XML representation of an object to a character column by means of a `COLMAP` clause in a `TABLE` or `MAP` statement.
- Oracle recommends the `AL32UTF8` character set as the database character set when working with XML data. This ensures the correct conversion by Oracle GoldenGate from source to target.
- Hierarchy-enabled tables are managed by the Oracle XML database repository and are supported with procedural logging.
- Assuming DDL support is enabled, Oracle GoldenGate replicates the CTAS statement and allows it to select the data from the underlying target table(s). This is controlled using the `TRANSLOGOPTIONS GETCTASDML` parameter and applies to all typed tables. If the parameter is used, the OIDs are preserved. For `XMLType` tables, the row object IDs must match between source and target, which cannot be maintained when Replicat uses logical SQL statements. `XMLType` tables created by an empty statement (that does not insert data in the new table) can be maintained correctly.
- `XMLType` tables with primary key-based object identifiers (OID)
- Relational tables with a single XML column
- SQL* Loader direct-path insert for `XML Binary` and `XML Object Relational`
- XML Schema-based `XMLType` are supported, but changes made to XML Schemas are not replicated and must be registered on both source and target databases with the `DBMS_XMLSCHEMA` package.
- Tables that contain `XMLType` columns must have at least one unique key constraint that is made up of scalar columns, or the combination of all scalar columns must guarantee uniqueness. Extract or Replicat cannot use unique or primary key constraints made up of XML attributes for row identification purposes.

1.6.8.2 Limitations of Support — Integrated Capture Mode

- `XML OR` and `XML Binary`, for native capture. XML binary/OR will be fetched if compatible with releases greater than 11.2.0.3.
- `XML CLOB`, source database compatibility is with releases greater than 11.0.0.0.0.
- The maximum length for the entire `SET` value of an update to an `XMLType` is 32K, including the new content plus other operators and `XQuery` bind values.

1.6.8.3 Limitations of Support — Classic Capture Mode

- For `XML Binary`, Oracle GoldenGate fetches additional row data from the source database. Because the fetched data may not part of the original transaction, it may lead to inconsistency.
- `XML Object Relational` is not supported in classic capture mode.

See Handling Special Data Types (page 10-3) for additional information about replicating XML.

1.6.9 User Defined or Abstract Types

Oracle GoldenGate supports User Defined types (UDT) or Abstract Data Types (ADT) when the source and target objects have the same structure. The schema names can be different.

- General Limitations of Support — Integrated and Classic Capture Modes (page 1-18)
- Limitations for Collection Types — Integrated and Classic Capture Modes (page 1-19)
- Limitations for Object Tables — Integrated and Classic Capture Modes (page 1-19)
- Limitations for Spatial Types — Integrated and Classic Capture Modes (page 1-20)

1.6.9.1 General Limitations of Support — Integrated and Classic Capture Modes

- Redo-based supports most attribute types, but falls back to fetching from source table when UDT contains:
  - Nested Table
  - `SDO_TOPO_GEOMETRY`
  - `SDO_GEORASTER`
- Fetch-based does not support UDT that contains:
  - `ANYDATA`
  - `TIMESTAMP WITH TIMEZONE`
  - `TIMESTAMP WITH LOCAL TIMEZONE`
  - `INTERVAL YEAR TO MONTH`
  - `INTERVAL DAY TO SECOND`
  - `BINARY FLOAT`
  - `BINARY DOUBLE`
  - `BFILE`
- Oracle GoldenGate does not support `UDTs` that contain:
  - `CFILE`
  - `REF`
  - `OPAQUE` (with exception of `XMLType` and `ANYDATA`:
- A table that contains a UDT must have one of the following: a primary key, column(s) with a unique constraint, or a unique index.
- Object or relational tables where the key contains a UDT, or where a UDT is the only column, are not supported.
- The `RMTTASK` parameter does not support UDT.

- CHAR and VARCHAR attributes that contain binary or unprintable characters are not supported.
- UDTs, including values inside object columns or rows, cannot be used within filtering criteria in TABLE or MAP statements, or as input or output for the Oracle GoldenGate column-conversion functions, SQLEXEC, or other built-in data-manipulation tools. Support is only provided for like-to-like Oracle source and targets.
- UDT and nested tables are supported with the following limitations:
  - Nested table UDTs cannot contain CHAR, NVARCHAR2, or NCLOB attributes.
  - Nested tables are not supported if there are extended (32k) VARCHAR2 or RAW attributes in UDTs.
  - Nested tables are not supported if there are CLOB or BLOB attributes in UDTs.
  - Nested table columns/attributes that are part of any other UDT are not supported.

1.6.9.2 Limitations for Collection Types — Integrated and Classic Capture Modes

- When data in a nested table is updated, the row that contains the nested table must be updated at the same time.
- When VARRAYS and nested tables are fetched, the entire contents of the column are fetched each time, not just the changes.

1.6.9.3 Limitations for Object Tables — Integrated and Classic Capture Modes

Integrated Capture Only (Redo-based)

- Redo-based captures object tables from redo when compatible with Oracle Database 12.2 and greater, but falls back to fetching from source table when an object table contains the following attributes:

  Nested table
  SDO_TOPO_GEOMETRY
  SDO_GEORASTER

  These objects are only fetched if they are compatible with Oracle GoldenGate 12.2.*x*.
- To fully support object tables created with CREATE TABLE as SELECT (CTAS) statement, Integrated Capture must be configured to capture DML from the CTAS statement. For more information about CTAS, see CREATE TABLE AS SELECT (page 13-23).
- An Oracle object table can be mapped to a non-Oracle object table in a supported target database.

Classic and Integrated Capture (Fetch-based)

- Fetch-based fetches all leaf-level attributes, as well as, root-level LOB, XML, UDT, ANYDATA, and collection attributes.
- Fetch-based does not support object tables that contain the following leaf-level attributes:

```
ANYDATA
TIMESTAMP WITH TIMEZONE
TIMESTAMP WITH LOCAL TIMEZONE
INTERVAL YEAR TO MONTH
INTERVAL DAY TO SECOND
BINARY FLOAT
BINARY DOUBLE
```

- Oracle GoldenGate supports object tables in uni-directional and active-active configurations. Object tables are captured from the redo log, but certain data types that are fetched from the database when in regular relational tables, such as LOBs and collection types, are also fetched when in object tables. Similarly, current limitations that apply to collection types when in regular tables also apply to these types when in object tables.
- A primary key must be defined on the root-level object attributes of the object table, and cannot include leaf-level attributes. If no key is defined, Oracle GoldenGate will use all useable columns as a pseudo-key.
- Oracle GoldenGate does not support the replication of DDL operations for an object table. This limitation includes the database object versioning that is associated with ALTERs of object tables.
- Synonyms are not supported for object tables or object types that contain object tables.

1.6.9.4 Limitations for Spatial Types — Integrated and Classic Capture Modes

Oracle GoldenGate supports SDO_GEOMETRY, SDO_TOPO_GEOMETRY, and SDO_GEORASTER (raster tables).

See additional configuration information in Handling Special Data Types in *Installing and Configuring Oracle GoldenGate for Oracle Database*.

1.6.10 Non-Supported Oracle Data Types

Oracle GoldenGate does not support the following data types.

- ANYDATA fetch-based column support for data types with VARRAYS that do not include named collections and VARRAYS embedded within those data types.
- Top-level VARRAY columns
- ANYDATASET
- ANYTYPE
- MLSLABEL
- ORDDICOM
- REFs
- TIMEZONE_ABBR
- URITYPE
- UDT containing an unsupported Oracle Data Type

- Oracle GoldenGate does not support replication of identity column data or Valid Time Temporal column data.

See additional exclusions in Summary of Supported Oracle Data Types and Objects Per Capture Mode (page 1-7).

1.7 Details of Support for Objects and Operations in Oracle DML

This section outlines the Oracle objects and operations that Oracle GoldenGate supports for the capture and replication of DML operations.

1.7.1 Multitenant Container Databases

Oracle GoldenGate captures from, and delivers to, a **multitenant container database**. See Configuring Oracle GoldenGate in a Multitenant Container Database (page 6-1).

1.7.1.1 Limitations for Multitenant Container Databases

Oracle GoldenGate Extract does not support Oracle Database 12.2 multitenant container databases.

1.7.2 Tables, Views, and Materialized Views

Oracle GoldenGate supports the following DML operations made to regular tables, index-organized tables, clustered tables, and materialized views.

- `INSERT`
- `UPDATE`
- `DELETE`
- Associated transaction control operations

> **Tip:**
>
> You can use the DBA_GOLDENGATE_SUPPORT_MODE data dictionary view to display information about the level of Oracle GoldenGate capture process support for the tables in your database. The PLSQL value of DBA_GOLDENGATE_SUPPORT_MODE indicates that the table is supported natively, but requires procedural supplemental logging. For more information, see the DBA_GOLDENGATE_SUPPORT_MODE. If you need to display all tables that have no primary and no non-null unique indexes, you can use the DBA_GOLDENGATE_NOT_UNIQUE. For more information, see DBA_GOLDENGATE_NOT_UNIQUE.

1.7.2.1 Limitations of Support for Regular Tables

These limitations apply to integrated and classic capture modes.

- Oracle GoldenGate supports tables that contain any number of rows.
- A row can be up to 4 MB in length. If Oracle GoldenGate is configured to include both the before and after image of a column in its processing scope, the 4 MB maximum length applies to the total length of the full before image plus the length of the after image. For example, if there are UPDATE operations on columns that are being used as a row identifier, the before and after images are processed and cannot exceed 4 MB in total. Before and after images are also required for columns that are not row identifiers but are used as comparison columns in conflict detection and resolution (CDR). Character columns that allow for more than 4 KB of data, such as a CLOB, only have the first 4 KB of data stored in-row and contribute to the 4MB maximum row length. Binary columns that allow for more than 4kb of data, such as a BLOB the first 8 KB of data is stored in-row and contributes to the 4MB maximum row length.
- Oracle GoldenGate supports the maximum number of columns per table that is supported by the database.
- Oracle GoldenGate supports the maximum column size that is supported by the database.
- Oracle GoldenGate supports tables that contain only one column, except when the column contains one of the following data types:
  - LOB
  - LONG
  - LONG VARCHAR
  - Nested table
  - UDT

  - VARRAY
  - XMLType
- Oracle GoldenGate supports tables with unused columns, but the support is disabled by default, and Extract abends on them. See Handling Other Database Properties (page 10-6).
- Oracle GoldenGate supports tables with these partitioning attributes:
  - Range partitioning
  - Hash Partitioning Interval Partitioning
  - Composite Partitioning
  - Virtual Column-Based Partitioning
  - Reference Partitioning
  - List Partitioning

  See Handling Other Database Properties (page 10-6).
- Oracle GoldenGate supports tables with virtual columns, but does not capture change data for these columns or apply change data to them: The database does not write virtual columns to the transaction log, and the Oracle Database does not permit DML on virtual columns. For the same reason, initial load data cannot be applied to a virtual column. You can map the data from virtual columns to non-virtual target columns. See Handling Other Database Properties (page 10-6).
- Oracle GoldenGate will not consider unique/index with virtual columns.
- Oracle GoldenGate supports replication to and from Oracle Exadata. To support Exadata Hybrid Columnar Compression, Extract must operate in integrated capture mode. To support Exadata Hybrid Columnar Compression, the source database compatibility must be set to 11.2.0.0.0 or higher.
- Oracle GoldenGate supports Transparent Data Encryption (TDE).
  - Extract supports TDE column encryption and TDE table space encryption without setup requirements in integrated capture mode. For integrated capture, the source database must be Oracle version 11.1.0 with compatibility setting of 11.0.0.0 or higher.
  - In classic capture mode, Extract supports column encryption for all versions of Oracle 11.1 and 11.2. Tablespace encryption is supported for all versions of Oracle 11.2.0.1. TDE in classic capture mode requires some setup. .
- Oracle GoldenGate supports `TRUNCATE` statements as part of its DDL replication support, or as standalone functionality that is independent of the DDL support. See Handling Other Database Properties (page 10-6).
- Oracle GoldenGate supports the capture of direct-load `INSERT`, with the exception of SQL*Loader direct-path insert for XML Binary and XML Object Relational as described in Limitations of Support — Integrated and Classic Capture Modes (page 1-17). Supplemental logging must be enabled, and the database must be in archive log mode. The following direct-load methods are supported.
  - `/*+ APPEND */` hint
  - `/*+ PARALLEL */` hint (Not supported for RAC in classic capture mode)
  - `SQLLDR` with `DIRECT=TRUE`

- Oracle GoldenGate fully supports capture from compressed objects when Extract is in integrated capture mode. The source database version must be 11.2.0.0 or higher if capturing from a downstream mining database or 11.2.0.3 if the source database is the mining database. Extract in classic capture mode does not support compressed objects.
- Oracle GoldenGate supports XA and PDML distributed transactions in integrated capture mode. Extract in classic capture mode does not support PDML or XA on RAC.
- Oracle GoldenGate supports DML operations on tables with `FLASHBACK ARCHIVE` enabled. However, Oracle GoldenGate does not support DDL that creates tables with the `FLASHBACK ARCHIVE` clause or DDL that creates, alters, or deletes the flashback data archive itself.

1.7.2.2 Limitations of Support for Index-Organized Tables

These limitations apply to classic capture mode.

- IOT with key compression enabled (indicated by the `COMPRESS` keyword in the `key_compression` clause) is not supported in classic capture mode, but is supported in integrated capture mode.

1.7.2.3 Limitations of Support for Views

These limitations apply to integrated and classic capture modes.

- Oracle GoldenGate supports capture from a view when Extract is in initial-load mode (capturing directly from the source view, not the redo log).
- Oracle GoldenGate does not capture change data from a view, but it supports capture from the underlying tables of a view.
- Oracle GoldenGate can replicate to a view as long as the view is inherently updateable. The structures of the source tables and a target view must be identical.

1.7.2.4 Limitations of Support for Materialized Views

Materialized views are supported by Extract in classic and integrated modes with the following limitations.

- Materialized views created `WITH ROWID` are not supported.
- The materialized view log can be created `WITH ROWID`.
- The source table must have a primary key.
- Truncates of materialized views are not supported. You can use a `DELETE FROM` statement.
- DML (but not DDL) from a full refresh of a materialized view is supported. If DDL support for this feature is required, open an Oracle GoldenGate support case.
- For Replicat the `Create MV` command must include the `FOR UPDATE` clause
- Either materialized views can be replicated or the underlying base table(s), but not both.

1.7.2.5 Limitations of Support for Clustered Tables

Indexed clusters are supported in both integrated and classic capture modes while hash clusters are not supported in either modes. In classic capture mode the following limitations apply:

- Encrypted and compressed clustered tables are not supported in classic capture.
- Extract in classic capture mode captures DML changes made to index clustered tables if the cluster size remains the same. Any DDL that causes the cluster size to increase or decrease may cause Extract to capture subsequent DML on that table incorrectly.

1.7.3 Sequences

- Oracle GoldenGate supports the replication of sequence values in a uni-directional and active-passive high-availability configuration.
- Oracle GoldenGate ensures that the target sequence values will always be higher than those of the source (or equal to them, if the cache is 0).
- Limitations of Support for Sequences (page 1-25)

1.7.3.1 Limitations of Support for Sequences

These limitations apply to integrated and classic capture modes.

- Oracle GoldenGate does not support the replication of sequence values in an active-active bi-directional configuration.
- The cache size and the increment interval of the source and target sequences must be identical. The cache can be any size, including 0 (`NOCACHE`).
- The sequence can be set to cycle or not cycle, but the source and target databases must be set the same way.

See configuration requirements in Handling Special Data Types in *Installing and Configuring Oracle GoldenGate for Oracle Database*.

1.7.4 Non-supported Objects and Operations in Oracle DML

The following are additional Oracle objects or operations that are not supported by Extract in either classic or integrated capture mode:

- `REF`
- Sequence values in an active-active bi-directional configuration
- Database Replay
- Tables created as `EXTERNAL`
- Invisible columns are not supported by classic Extract

The following are not supported in classic capture mode:

- Exadata Hybrid Columnar Compression
- Capture from tables with OLTP table compression

- Capture from tablespaces and tables created or altered with `COMPRESS`
- Capture from encrypted and compressed clustered tables
- Invisible column
- Distributed transactions. In Oracle versions 11.1.0.6 and higher, you can capture these transactions if you make them non-distributed by using the following command, which requires the database to be restarted.

  ```
  alter system set _CLUSTERWIDE_GLOBAL_TRANSACTIONS=FALSE;
  ```
- RAC distributed XA and PDML distributed transactions
- Version enabled-tables
- Identity columns

1.8 Details of Support for Objects and Operations in Oracle DDL

This section outlines the Oracle objects and operation types that Oracle GoldenGate supports for the capture and replication of DDL operations. Trigger-based capture is required for Oracle releases that are earlier than version 11.2.0.4. If Extract will run in integrated mode against a version 11.2.0.4 or later Oracle Database, the DDL trigger and supporting objects are not required.

- Supported Objects and Operations in Oracle DDL (page 1-26)
- Non-supported Objects and Operations in Oracle DDL (page 1-28)

1.8.1 Supported Objects and Operations in Oracle DDL

When the source database is Oracle 11.2.0.4 or later and Extract operates in integrated mode, DDL capture support is integrated into the database logmining server and does not require the use of a DDL trigger. You must set the database parameter compatibility to 11.2.0.4.0. In integrated capture mode, Extract supports DDL that includes password-based column encryption, such as:

- `CREATE TABLE t1 (a number, b varchar2(32) ENCRYPT IDENTIFIED BY my_password);`
- `ALTER TABLE t1 ADD COLUMN c varchar2(64) ENCRYPT IDENTIFIED BY my_password;`

> **Note:**
>
> Password-based column encryption in DDL is not supported in classic capture mode.

The following additional statements apply to both integrated and classic capture modes with respect to DDL support.

- All Oracle GoldenGate topology configurations are supported for Oracle DDL replication.
- Active-active (bi-directional) replication of Oracle DDL is supported between two (and only two) databases that contain identical metadata.

- Oracle GoldenGate supports DDL on the following objects:
  - clusters
  - directories
  - functions
  - indexes
  - packages
  - procedure
  - tables
  - tablespaces
  - roles
  - sequences
  - synonyms
  - triggers
  - types
  - views
  - materialized views
  - users
  - invisible columns
- Oracle Edition-Based Redefinition (EBR) database replication of Oracle DDL is supported for integrated Extract for the following Oracle Database objects:
  - functions
  - library
  - packages (specification and body)
  - procedure
  - synonyms
  - types (specification and body)
  - views

  EBR does not support use of DDL triggers.
- Oracle GoldenGate supports DDL operations of up to 4 MB in size. Oracle GoldenGate measures the size of DDL statement in bytes, not in characters. This size limitation includes packages, procedures, and functions. The actual size limit of the DDL support is approximate, because the size not only includes the statement text, but also Oracle GoldenGate maintenance overhead that depends on the length of the object name, the DDL type, and other characteristics of keeping a DDL record internally.
- Oracle GoldenGate supports Global Temporary Tables (GTT) DDL operations to be visible to Extract so that they can be replicated. You must set the `DDLOPTIONS` parameter to enable this operation because it is not set by default.
- Oracle GoldenGate supports Integrated Dictionary for use with `NOUSERID` and `TRANLOGOPTIONS GETCTASDML`. This means that Extract will be obtaining object

metadata from the LogMiner dictionary instead of the DDL trigger and without querying the dictionary objects. Oracle GoldenGate uses Integrated Dictionary automatically when the source database compatibility parameter is greater than or equal to 11.2.0.4 and Integrated Extract is used.

The Integrated Dictionary feature is *not* supported with classic Extract.

When using Integrated Dictionary and trail format in the Oracle GoldenGate release 12.2.*x*, Integrated Capture requires the Logminer patch to be applied on the mining database if the Oracle Database release is earlier than 12.1.0.2.

- Oracle GoldenGate supports replication of invisible columns in Integrated Capture mode. Trail format release 12.2 is required. Replicat must specify the `MAPINVISIBLECOLUMNS` parameter or explicitly map to invisible columns in the `COLMAP` clause of the `MAP` parameter.

  If `SOURCEDEFS` or `TARGETDEFS` is used, the metadata format of a definition file for Oracle tables must be compatible with the trail format. Metadata format 12.2 is compatible with trail format 12.2, and metadata format earlier than 12.2 is compatible with trail format earlier than 12.2. To specify the metadata format of a definition file, use the `FORMAT RELEASE` option of the `DEFSFILE` parameter when the definition file is generated in DEFGEN.
- DDL statements to create a namespace context (`CREATE CONTEXT`) are captured by Extract and applied by Replicat.
- Extract in pump mode supports the following DDL options:
  - `DDL INCLUDE ALL`
  - `DDL EXCLUDE ALL`
  - `DDL EXCLUDE OBJNAME`

  The `SOURCECATALOG` and `ALLCATALOG` option of `DDL EXCLUDE` is also supported.

  If no DDL parameter is specified, then all DDLs are written to trail. If `DDL EXCLUDE OBJNAME` is specified and the object owner is does not match an exclusion rule, then it is written to the trail.

1.8.2 Non-supported Objects and Operations in Oracle DDL

These statements apply to integrated and classic capture modes.

- Excluded Objects (page 1-28)
- Other Non-supported DDL (page 1-30)

1.8.2.1 Excluded Objects

The following names or name prefixes are considered Oracle-reserved and must be excluded from the Oracle GoldenGate DDL configuration. Oracle GoldenGate will ignore objects that contain these names.

Excluded schemas:

```
"ANONYMOUS", // HTTP access to XDB
"APPQOSSYS", // QOS system user
"AUDSYS", // audit super user
"BI", // Business Intelligence
"CTXSYS", // Text
"DBSNMP", // SNMP agent for OEM
```

```
"DIP", // Directory Integration Platform
"DMSYS", // Data Mining
"DVF", // Database Vault
"DVSYS", // Database Vault
"EXDSYS", // External ODCI System User
"EXFSYS", // Expression Filter
"GSMADMIN_INTERNAL", // Global Service Manager
"GSMCATUSER", // Global Service Manager
"GSMUSER", // Global Service Manager
"LBACSYS", // Label Security
"MDSYS", // Spatial
"MGMT_VIEW", // OEM Database Control
"MDDATA",
"MTSSYS", // MS Transaction Server
"ODM", // Data Mining
"ODM_MTR", // Data Mining Repository
"OJVMSYS", // Java Policy SRO Schema
"OLAPSYS", // OLAP catalogs
"ORACLE_OCM", // Oracle Configuration Manager User
"ORDDATA", // Intermedia
"ORDPLUGINS", // Intermedia
"ORDSYS", // Intermedia
"OUTLN", // Outlines (Plan Stability)
"SI_INFORMTN_SCHEMA", // SQL/MM Still Image
"SPATIAL_CSW_ADMIN", // Spatial Catalog Services for Web
"SPATIAL_CSW_ADMIN_USR",
"SPATIAL_WFS_ADMIN", // Spatial Web Feature Service
"SPATIAL_WFS_ADMIN_USR",
"SYS",
"SYSBACKUP",
"SYSDG",
"SYSKM",
"SYSMAN", // Adminstrator OEM
"SYSTEM",
"TSMSYS", // Transparent Session Migration
"WKPROXY", // Ultrasearch
"WKSYS", // Ultrasearch
"WK_TEST",
"WMSYS", // Workspace Manager
"XDB", // XML DB
"XS$NULL",
"XTISYS", // Time Index
```

Special schemas:

```
"AURORA$JIS$UTILITY$", // JSERV
"AURORA$ORB$UNAUTHENTICATED", // JSERV
"DSSYS", // Dynamic Services Secured Web Service
"OSE$HTTP$ADMIN", // JSERV
"PERFSTAT", // STATSPACK
"REPADMIN",
"TRACESVR" // Trace server for OEM
```

Excluded tables (the * wildcard indicates any schema or any character):

```
"*.AQ$*", // advanced queues
"*.DR$*$*", // oracle text
"*.M*_*$$", // Spatial index
"*.MLOG$*", // materialized views
"*.OGGQT$*",
"*.OGG$*", // AQ OGG queue table
```

```
"*.ET$*", // Data Pump external tables
"*.RUPD$*", // materialized views
"*.SYS_C*", // constraints
"*.MDR*_*$", // Spatial Sequence and Table
"*.SYS_IMPORT_TABLE*",
"*.CMP*$*", // space management, rdbms >= 12.1
"*.DBMS_TABCOMP_TEMP_*", // space management, rdbms < 12.1
"*.MDXT_*$*" // Spatial extended statistics tables
```

1.8.2.2 Other Non-supported DDL

Oracle GoldenGate does not support the following:

- DDL on nested tables.
- DDL on identity columns.
- `ALTER DATABASE` and `ALTER SYSTEM` (these are not considered to be DDL) When using Integrated Dictionary you can replicate `ALTER DATABASE DEFAULT EDITION` and `ALTER PLUGGABLE DATABASE DEFAULT EDITION`. All other `ALTER [PLUGABLE] DATABASE` commands are ignored.
- DDL on a standby database.
- Database link DDL.
- DDL that creates tables with the `FLASHBACK ARCHIVE` clause and DDL that creates, alters, or deletes the flashback data archive itself. DML on tables with `FLASHBACK ARCHIVE` is supported.
- Classic capture mode does not support DDL that includes password-based column encryption, such as:
  - `CREATE TABLE t1 (a number, b varchar2(32) ENCRYPT IDENTIFIED BY my_password);`
  - `ALTER TABLE t1 ADD COLUMN c varchar2(64) ENCRYPT IDENTIFIED BY my_password;`

1.9 Supported and Non-supported Object Names

Oracle object names are case insensitive by default, but can be made case-sensitive with the use of double quotes. Oracle GoldenGate supports Oracle case-sensitivity. For information about Oracle GoldenGate support for object names and case, see *Administering Oracle GoldenGate for Windows and UNIX*.

2
Installing Classic Oracle GoldenGate

This chapter includes the instructions for installing Oracle GoldenGate for the first time. Installing Oracle GoldenGate installs all of the components that are required to run and manage the processing (excluding any components required from other vendors, such as drivers or libraries) and it installs the Oracle GoldenGate utilities. This chapter includes the following sections:

These instructions are for installing Oracle GoldenGate for the first time. Additionally, they are for downloading the base release of a new version of Oracle GoldenGate.

To download and install subsequent patches to the base release, go to the Patches and Updates tab of My Oracle Support at:

`http://support.oracle.com`

To upgrade Oracle GoldenGate from one version to another, follow the upgrade instructions at:

`http://docs.oracle.com/goldengate/c1221/gg-winux/index.html`

- Understanding and Obtaining the Oracle GoldenGate Distribution (page 2-1)
- Setting ORACLE_HOME and ORACLE_SID (page 2-2)
- Setting Library Paths for Dynamic Builds on UNIX (page 2-3)
- Preparing to Install Oracle GoldenGate Within a Cluster (page 2-5)
- Installing Oracle GoldenGate (page 2-6)
- Integrating Oracle GoldenGate into a Cluster (page 2-10)

2.1 Understanding and Obtaining the Oracle GoldenGate Distribution

For complete information about how to obtain Oracle Fusion Middleware software, see "Understanding and Obtaining Product Distributions" in *Planning an Installation of Oracle Fusion Middleware*.

To download the Oracle GoldenGate software for development or evaluation, see the following location on the Oracle Technology Network (OTN):

http://www.oracle.com/technetwork/middleware/fusion-middleware/downloads/index.html

For more information about locating and downloading Oracle Fusion Middleware products, see the *Oracle Fusion Middleware Download, Installation, and Configuration Readme Files* on OTN.

To obtain Oracle GoldenGate follow these steps:

1. Go to Oracle Technology Network.
2. Find the Oracle GoldenGate 12*c* (12.2.0.1) release and download the ZIP file onto your system.

2.2 Setting ORACLE_HOME and ORACLE_SID

Make certain that the ORACLE_HOME and ORACLE_SID system environment variables are set to the correct Oracle instance. The Oracle GoldenGate processes refer to them when connecting to the database.

Specifying Oracle Variables on UNIX and Linux Systems (page 2-2)

Specifying Oracle Variables on Windows Systems (page 2-3)

- Specifying Oracle Variables on UNIX and Linux Systems (page 2-2)
- Specifying Oracle Variables on Windows Systems (page 2-3)

2.2.1 Specifying Oracle Variables on UNIX and Linux Systems

If there is one instance of Oracle Database on the system, set the ORACLE_HOME and ORACLE_SID environment variables at the system level. If you cannot set them that way, use the following SETENV statements in the parameter file of every Extract and Replicat group that will be connecting to the instance. The SETENV parameters override the system settings and allow the Oracle GoldenGate process to set the variables at the session level when it connects to the database.

```
SETENV (ORACLE_HOME = "path to Oracle home location")

SETENV (ORACLE_SID = "SID")
```

If there are multiple Oracle instances on the system with Extract and Replicat processes connecting to them, you will need to use a SETENV statement in the parameter file of each process group. As input to the SETENV parameter, use the ORACLE_HOME and ORACLE_SID environment variables to point Oracle GoldenGate to the correct Oracle instance. For example, the following shows parameter files for two Extract groups, each capturing from a different Oracle instance.

Group 1:

```
EXTRACT ora9a
SETENV (ORACLE_HOME = "/home/oracle/ora/product")
SETENV (ORACLE_SID = "oraa")
USERIDALIAS tiger1
RMTHOST sysb
RMTTRAIL /home/ggs/dirdat/rt
TABLE hr.emp;
TABLE hr.salary;
```

Group 2:

```
EXTRACT orab
SETENV (ORACLE_HOME = "/home/oracle/ora/product")
SETENV (ORACLE_SID = "orab")
USERIDALIAS tiger1
RMTHOST sysb
RMTTRAIL /home/ggs/dirdat/st
TABLE fin.sales;
TABLE fin.cust;
```

2.2.2 Specifying Oracle Variables on Windows Systems

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.

1. On the Desktop or Start menu (depending on the Windows version), right-click **My Computer**, and then select **Properties**.
2. In Properties, click the **Advanced** tab.
3. Click **Environment Variables**.
4. Under System Variables, click **New**.
5. For Variable Name, type `ORACLE_HOME`.
6. For Variable Value, type the path to the Oracle binaries.
7. Click **OK**.
8. Click **New** again.
9. For Variable Name, type `ORACLE_SID`.
10. For Variable Value, type the instance name.
11. Click **OK**.

If there are multiple Oracle instances on the system with Extract and Replicat processes connecting to them, do the following.

1. Use the preceding procedure (single Oracle instance on system) to set the `ORACLE_HOME` and `ORACLE_SID` system variables to the first Oracle instance.
2. Start all of the Oracle GoldenGate processes that will connect to that instance.
3. Repeat the procedure for the next Oracle instance, but first edit the existing `ORACLE_HOME` and `ORACLE_SID` variables to specify the new information.
4. Start the Oracle GoldenGate processes that will connect to that instance.
5. Repeat the edit and startup procedure for the rest of the Oracle instances.

2.3 Setting Library Paths for Dynamic Builds on UNIX

Oracle GoldenGate uses shared libraries. When you install Oracle GoldenGate on a UNIX system, the following must be true *before you run GGSCI or any other Oracle GoldenGate process.*

1. Make certain that the database libraries are added to the shared-library environment variables of the system. This procedure is usually performed at

database installation time. Consult your Database Administrator if you have any questions.

When Oracle GoldenGate is running on the same server as the database, all of the following must be 64-bit:

- Oracle library versions
- Oracle GoldenGate version
- Database versions

When Oracle GoldenGate connects remotely to the database server through SQL*Net, the following are required:

- Replicat: The Oracle client library and the Oracle GoldenGate build must have the same Oracle version, bit type (64-bit or IA64), and operating system version.
- Extract: The Oracle client library and the Oracle GoldenGate build must have the same Oracle version, bit type (64-bit or IA64), and operating system version. In addition, both operating systems must be the same endian.

2. If you will be running an Oracle GoldenGate program from outside the Oracle GoldenGate installation directory on a UNIX system:
   - (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.

   For example, given an Oracle GoldenGate installation directory of `/users/ogg`, the second command in the following example requires these variables to be set:

Command	Requires GG libraries in environment variable?
`$ users/ogg > ./ggsci`	No
`$ users > ./ogg/ggsci`	Yes

Example 2-1 To Set the Variables in Korn Shell

```
PATH=installation_directory:$PATH
export PATH
shared_libraries_variable=absolute_path_of_installation_directory:$shared_libraries_variable
export shared_libraries_variable
```

Example 2-2 To Set the Variables in Bourne Shell

```
export PATH=installation_directory:$PATH
export
shared_libraries_variable=absolute_path_of_installation_directory:$shared_libraries_variable
```

Example 2-3 To Set the Variables in C Shell

```
setenv PATH installation_directory:$PATH
setenv shared_libraries_variable
absolute_path_of_installation_directory:$shared_libraries_variable
```

Where *shared libraries variable* is one of the variables shown in Table 2-1 (page 2-5):

Table 2-1 UNIX/Linux Library Path Variables Per Platform

Platform	Environment variable
IBM AIX	LIBPATH
HP-UX	LD_LIBRARY_PATH
Sun Solaris LINUX	LD_LIBRARY_PATH[1]

[1] In 64-bit environments with 32-bit Oracle Databases, Oracle GoldenGate requires the LD_LIBRARY_PATH to include the 32-bit Oracle libraries.

The following is an example of how to set the path in Bourne shell:

```
export LD_LIBRARY_PATH=/ggs/11.0:$LD_LIBRARY_PATH
```

> **Note:**
>
> To view the libraries that are required by an Oracle Oracle GoldenGate process, use the `ldd goldengate_process` shell command before starting the process. This command also shows an error message for any that are missing.

2.4 Preparing to Install Oracle GoldenGate Within a Cluster

This topic covers the installation requirements that apply when Oracle GoldenGate will be installed in a cluster environment. Oracle GoldenGate can be used with any cluster-management solution that has the ability to automate failover. The Oracle Clusterware solution provides the advantage of being able to be used with or without an Oracle RAC database, which enables you to include any non-database servers that are running Oracle GoldenGate.

- Deciding Where to Install Oracle GoldenGate Binaries and Files in the Cluster (page 2-5)
- Example Oracle Cluster Storage (page 2-6)

2.4.1 Deciding Where to Install Oracle GoldenGate Binaries and Files in the Cluster

To ensure High Availability with Oracle GoldenGate it is recommended install the GoldenGate binaries on local storage for each node in the cluster. With a local node installation it is possible to patch and upgrade the GoldenGate software on one or more nodes in the cluster without affecting the node where GoldenGate is currently running. Then at a predetermined time, GoldenGate can be switched to one of the upgraded nodes.

To enable GoldenGate to run on any of the clustered nodes there is a requirement to place certain directories on shared storage. This provides the ability of the GoldenGate

processes to restart from its last known position when running on a different node in the cluster.

When running Oracle GoldenGate in this manner follow these recommendations:

- The Oracle GoldenGate installation must have the same location path on every node.
- At minimum, install the following directories on the shared storage to support Oracle GoldenGate recovery requirements. On UNIX or Linux, you can create symbolic links to them from the installation directory on each node.
  - `br`
  - `dirchk`
  - `dirdat`
  - `dirbdb`
  - `dirprm`

  These directories are among those created when you issue `CREATE SUBDIRS` during installation.

See Integrating Oracle GoldenGate into a Cluster (page 2-10) after you install Oracle GoldenGate.

For more information about installing and using Oracle GoldenGate in a cluster, see the Oracle Maximum Availability (MAA) white paper: `http://www.oracle.com/technetwork/database/features/availability/maa-goldengate-rac-2007111.pdf`.

2.4.2 Example Oracle Cluster Storage

You will need to install at least some Oracle GoldenGate directories on shared storage. Select cluster-aware shared storage that is independent of, but available to, all nodes of the cluster. You could use any of the following example Oracle cluster storage options:

- Oracle Database File System (DBFS) - Database File System (DBFS) creates a standard file system interface on top of files and directories that are stored in database tables. DBFS is similar to NFS in that it provides a shared network file system that looks like a local file system and has both a server component and a client component. Because the files are stored inside the database they are protected using the database backup and recovery features, along with High Availability using Oracle Data Guard.
- Oracle Storage Management Cluster File System (ACFS) - Oracle ACFS is a multi-platform, scalable file system, and storage management technology that extends Oracle Automatic Storage Management (Oracle ASM) functionality to provide a POSIX compatible file system.
- Oracle ZFS - Using an Oracle ZFS Appliance that can be mounted from each node in the cluster using the NFS protocol.

2.5 Installing Oracle GoldenGate

Oracle GoldenGate for Oracle Database is installed from the Oracle Universal Installer (OUI). The OUI is a graphic installation program that prompts you for the input

required to install the Oracle GoldenGate binaries and working files, and set the correct database environment in which Oracle GoldenGate will operate.

You can use OUI on any of the Linux, UNIX, and Windows platforms that OUI supports and which Oracle GoldenGate supports.

OUI is supported for Oracle versions 11g and later. An instance of Oracle GoldenGate can be installed for only one major Oracle Database version in any given Oracle home. For example, if you have Oracle Database 11.2 and 12.1, you must have separate Oracle GoldenGate installations for each one. This does not apply to data patch levels within the same major release. You can install multiple instances of Oracle GoldenGate for the same or different database versions on the same host.

The installer registers the Oracle GoldenGate home directory with the central inventory that is associated with the selected database. The inventory stores information about all Oracle software products installed on a host, provided the product was installed using OUI.

- Performing an Interactive Installation with OUI (page 2-7)
- Performing a Silent Installation with OUI (page 2-8)

2.5.1 Performing an Interactive Installation with OUI

The interactive installation provides a graphical user interface that prompts for the required installation information. These instructions apply to new installations as well as upgrades. However, to perform an upgrade to Oracle GoldenGate, follow the instructions in *Upgrading Oracle GoldenGate for Windows and UNIX*, which includes a prompt to run OUI at the appropriate time.

1. Expand the installation file.
2. From the expanded directory, run the `runInstaller` program on UNIX or Linux, or run `setup.exe` on Windows.
3. On the **Select Installation Option** page, select the Oracle GoldenGate version to install, and then click **Next** to continue.
4. On the **Specify Installation Details** page, specify the following:
   - For **Software Location**, specify the Oracle GoldenGate installation directory. It can be a new or existing directory (for any Oracle GoldenGate version prior to 12.1.2.0.0) that is empty and has the amount of disk space shown on the screen or in the existing Oracle GoldenGate installation location (if you are upgrading an existing Oracle GoldenGate installation). The default location is under the installing user's home directory, but Oracle recommends changing it to a local directory that is not mounted and has no quotas. The specified directory cannot be a registered home in the Oracle central inventory. If installing in a cluster, install Oracle GoldenGate on local storage on each node in the cluster to provide high availability options for upgrading and software patching. See Preparing to Install Oracle GoldenGate Within a Cluster (page 2-5) for more information about installing Oracle GoldenGate in a cluster.
   - (Optional) Select **Start Manager** to perform configuration functions, such as creating the Oracle GoldenGate subdirectories in the installation location, setting library paths, and starting Manager on the specified port number. To proceed, a database must exist on the system. When **Start Manager** is selected, the **Database Location and Manager Port** fields are displayed.

  - For **Database Location**, the database version in the specified location must be **Oracle Database 12c** if you are installing Oracle GoldenGate for Oracle Database 12c or **Oracle Database 11g** if you are installing Oracle GoldenGate for Oracle Database11g. The database must have a registered home in the Oracle central inventory. The installer registers the Oracle GoldenGate home directory with the central inventory.
  - For **Manager Port**, accept the default port number or enter a different unreserved, unrestricted port number for the Manager process to use for interprocess communication. The default port is the first available one starting with 7809. If you are installing multiple instances of Oracle GoldenGate on the same system, each must use a different port number.

- Click **Next** to continue. If this is an upgrade to an existing Oracle GoldenGate installation, OUI prompts that the selected software location has files or directories. Click **Yes**.

5. The **Create Inventory** page is displayed if this is the first Oracle product to be installed from OUI on a host that does not have a central inventory.
   - For **Inventory Directory**, specify a directory for the central inventory. It can be a new directory or an existing directory that is empty and has the amount of disk space shown on the screen. The directory cannot be on a shared drive.
   - Select an operating system group in which the members have write permission to the inventory directory. This group is used to add inventory information to the Oracle GoldenGate subfolder.
6. On the **Summary** page, confirm that there is enough space for the installation and that the installation selections are correct. Optionally, click **Save Response File** to save the installation information to a response file. You can run the installer from the command line with this file as input to duplicate the results of a successful installation on other systems. You can edit this file or create a new one from a template. See Performing a Silent Installation with OUI (page 2-8).
7. Click **Install** to begin the installation or **Back** to go back and change any input specifications. When upgrading an existing Oracle GoldenGate installation, OUI notifies you that the software location has files or directories. Click **Yes** to continue. You are notified when the installation is finished.
8. If you created a central inventory directory, you are prompted to run the `INVENTORY_LOCATION/orainstRoot.sh` script. This script must be executed as the root operating system user. This script establishes the inventory data and creates subdirectories for each installed Oracle product (in this case, Oracle GoldenGate).

2.5.2 Performing a Silent Installation with OUI

These instructions apply to new installations as well as upgrades. However, to perform an upgrade to Oracle GoldenGate, follow the instructions in Overview of Upgrading Oracle GoldenGate, which include a prompt to run OUI at the appropriate time.

You can perform a silent installation from the command console if the system has no X-Windows interface or to perform an automated installation. Silent installations can ensure that multiple users in your organization use the same installation options when they install your Oracle products.

You perform a silent installation by running a response file. You can create a response file by selecting the **Save Response File** option during an interactive OUI session or

by editing a template, as shown in Example 2-4 (page 2-6). To run a response file, issue the following command.

```
-silent -nowait -responseFile path_to_file
```

The response files and the template are stored in the response sub-directory of the Oracle GoldenGate installation directory. The Oracle GoldenGate response file contains a standard set of Oracle configuration parameters in addition to parameters that are specific to Oracle GoldenGate. These parameters correspond to the fields in the interactive session.

> **Note:**
>
> If you are upgrading an existing Oracle GoldenGate installation with the silent option, you might get the following warning:
>
> WARNING:OUI-10030:You have specified a non-empty directory to install this product. It is recommended to specify either an empty or a non-existent directory. You may, however, choose to ignore this message if the directory contains Operating System generated files or subdirectories like lost +found.Do you want to proceed with installation in this Oracle Home?
>
> Press **ENTER** to continue.

Example 2-4 Oracle GoldenGate Response File Template

```
###################################################################
## Copyright(c) Oracle Corporation 2013. All rights reserved.     ##
##                                                               ##
## Specify values for the variables listed below to customize    ##
## your installation.                                            ##
##                                                               ##
## Each variable is associated with a comment. The comment       ##
## can help to populate the variables with the appropriate       ##
## values.                                                       ##
##                                                               ##
## IMPORTANT NOTE: This file should be secured to have read      ##
## permission only by the oracle user or an administrator who    ##
## own this installation to protect any sensitive input values.  ##
##                                                               ##
###################################################################
#------------------------------------------------------------------------------
# Do not change the following system generated value.
#------------------------------------------------------------------------------
oracle.install.responseFileVersion=/oracle/install/rspfmt_ogginstall_response_
schema_v12_1_2###############################################################################
#########
##                                                                           ##
## Oracle GoldenGate installation option and details
####                                                                          ##
###############################################################################
#------------------------------------------------------------------------------
# Specify the installation option.
# Specify ORA12c for installing Oracle GoldenGate for Oracle Database 12c and
#         ORA11g for installing Oracle GoldenGate for Oracle Database 11g
#------------------------------------------------------------------------------
INSTALL_OPTION=
```

```
#-------------------------------------------------------------------------------
# Specify a location to install Oracle
GoldenGate#-------------------------------------------------------------------------------
-----
SOFTWARE_LOCATION=

#-------------------------------------------------------------------------------
# Specify true to start the manager after installation.

#-------------------------------------------------------------------------------
START_MANAGER=

#-------------------------------------------------------------------------------
# Specify a free port within the valid range for the manager process.
# Required only if START_MANAGER is true.
#-------------------------------------------------------------------------------
MANAGER_PORT=

#-------------------------------------------------------------------------------
# Specify the location of the Oracle Database.
# Required only if START_MANAGER is true.
#-------------------------------------------------------------------------------
DATABASE_LOCATION=

###############################################################################
##                                                                           ##
## Specify details to Create inventory for Oracle installs                   ##
## Required only for the first Oracle product install on a system.           ##
##                                                                           ##
###############################################################################

#-------------------------------------------------------------------------------
# Specify the location which holds the install inventory files.
# This is an optional parameter if installing on# Windows based Operating
System.#-------------------------------------------------------------------------------
--INVENTORY_LOCATION=

#-------------------------------------------------------------------------------
# UNIX group to be set for the inventory directory.  # This parameter is not
applicable if installing on
# Windows based Operating System.
#-------------------------------------------------------------------------------
UNIX_GROUP_NAME=
```

2.6 Integrating Oracle GoldenGate into a Cluster

If you installed Oracle GoldenGate in a cluster, take the following steps to integrate Oracle GoldenGate within the cluster solution.

For more information about installing and using Oracle GoldenGate in a cluster, see the Oracle GoldenGate with Oracle Real Application Clusters Configuration white paper http://www.oracle.com/technetwork/database/features/availability/maa-goldengate-rac-2007111.pdf.

- General Requirements in a Cluster (page 2-11)
- Adding Oracle GoldenGate as a Windows Cluster Resource (page 2-11)

2.6.1 General Requirements in a Cluster

1. Configure the Oracle Grid Infrastructure Bundled Agent (XAG) to automatically manage the GoldenGate processes on the cluster nodes. Using the XAG will make sure the required cluster file system is mounted before the GoldenGate processes are started. If an application virtual IP (VIP) is used in the cluster the bundled agent will also ensure the VIP is started on the correct node.
2. Configure the Oracle GoldenGate Manager process with the `AUTOSTART` and `AUTORESTART` parameters so that Manager starts the replication processes automatically.
3. Mount the shared drive on one node only. This prevents processes from being started on another node. Use the same mount point on all nodes. If you are using the Oracle Grid Infrastructure Bundled Agent, the mounting of the required file systems are automatically carried out.
4. Ensure that all database instances in the cluster have the same `COMPATIBLE` parameter setting.
5. Configure Oracle GoldenGate as directed in this documentation.

2.6.2 Adding Oracle GoldenGate as a Windows Cluster Resource

When installing Oracle GoldenGate in a Windows cluster, follow these instructions to establish Oracle GoldenGate as a cluster resource and configure the Manager service correctly on all nodes.

- In the cluster administrator, add the Manager process to the group that contains the database instance to which Oracle GoldenGate will connect.
- Make sure all nodes on which Oracle GoldenGate will run are selected as possible owners of the resource.
- Make certain the Manager Windows service has the following dependencies (can be configured from the Services control panel):
  - The database resource
  - The disk resource that contains the Oracle GoldenGate directory
  - The disk resource that contains the database transaction log files
  - The disk resource that contains the database transaction log backup files

3

Preparing the Database for Oracle GoldenGate

This chapter contains steps to take so that the source Oracle Database is configured properly to support the capture of transactional changes.

Topics:

3.1 Configuring Connections for Integrated Processes

If you will be using integrated capture and integrated Replicat, each requires a dedicated server connection in the `tnsnames.ora` file. You direct the processes to use these connections with the `USERID` or `USERIDALIAS` parameter in the Extract and Replicat parameter files when you configure those processes.

The following is an example of the dedicated connection required for integrated capture (Extract) and integrated Replicat.

```
TEST =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = test2)(PORT = 1521))
    )
(CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = test)
  )
)
```

The following are the security options for specifying the connection string in the Extract or Replicat parameter file.

Password encryption method:

```
USERID intext@test, PASSWORD mypassword
```

Credential store method:

```
USERIDALIAS ext
```

In the case of `USERIDALIAS`, the alias `ext` is stored in the Oracle GoldenGate credential store with the actual connection string, as in the following example:

```
GGSCI> INFO CREDENTIALSTORE DOMAIN support
Domain: Support
```

```
Alias: ext
Userid: intext@test
```

For more information about specifying database connection information in the parameter file, see *Administering Oracle GoldenGate for Windows and UNIX*.

3.2 Configuring Logging Properties

Oracle GoldenGate relies on the redo logs to capture the data that it needs to replicate source transactions. The Oracle redo logs on the source system must be configured properly before you start Oracle GoldenGate processing.

> **Note:**
>
> Redo volume is increased as the result of this required logging. You can wait until you are ready to start Oracle GoldenGate processing to enable the logging.

This section addresses the following logging levels that apply to Oracle GoldenGate. Which logging level that you use is dependent on the Oracle GoldenGate feature or features that you are using.

- Enabling Minimum Database-level Supplemental Logging (page 3-4)
- Enabling Schema-level Supplemental Logging (page 3-5)
- Enabling Table-level Supplemental Logging (page 3-7)

Table 3-1 (page 3-2) shows the Oracle GoldenGate use cases for the different logging properties. Detailed information follows.

Table 3-1 Supplemental Logging Options Per Use Case

Logging option	GGSCI command	What it does	Use case
Forced logging mode	None; enable through the database.	Forces the logging of all transactions and loads.	Strongly recommended for all Oracle GoldenGate use cases.
Minimum database-level supplemental logging	None; enable through the database.	Enables minimal supplemental logging to add row-chaining information to the redo log.	Required for all Oracle GoldenGate use cases
Schema-level supplemental logging, default setting See Enabling Schema-level Supplemental Logging (page 3-5).	`ADD SCHEMATRANDATA`	Enables unconditional supplemental logging of the primary key and conditional supplemental logging of unique key(s) and foreign key(s) of all tables in a schema. All of these keys together are known as the *scheduling columns*.	Enables the logging for all current and future tables in the schema. If the primary key, unique key, and foreign key columns are not identical at both source and target, use `ALLCOLS`. Required when using DDL support.

Table 3-1 (Cont.) Supplemental Logging Options Per Use Case

Logging option	GGSCI command	What it does	Use case
Schema-level supplemental logging with unconditional logging for all supported columns. (See Enabling Schema-level Supplemental Logging (page 3-5) for non-supported column types.)	`ADD SCHEMATRANDATA` with `ALLCOLS` option	Enables unconditional supplemental logging of all of the columns in a table, for all of the tables in a schema.	Used for bidirectional and active-active configurations where all column values are checked, not just the changed columns, when attempting to perform an update or delete. This takes more resources though allows for the highest level of real-time data validation and thus conflict detection. It can also be used when the source and target primary, unique, and foreign keys are not the same or are constantly changing between source and target.
Schema-level supplemental logging, minimal setting	`ADD SCHEMATRANDATA` with `NOSCHEDULINGCOLS` option	Enables unconditional supplemental logging of the primary key and all valid unique indexes of all tables in a schema.	Use only for nonintegrated Replicat. This is the minimum required schema-level logging.
Table-level supplemental logging with built-in support for integrated Replicat See Enabling Table-level Supplemental Logging (page 3-7)	`ADD TRANDATA`	Enables unconditional supplemental logging of the primary key and conditional supplemental logging of unique key(s) and foreign key(s) of a table. All of these keys together are known as the *scheduling columns*.	Required for all Oracle GoldenGate use cases unless schema-level supplemental logging is used. If the primary key, unique key, and foreign key columns are not identical at both source and target, use `ALLCOLS`.

Table 3-1 (Cont.) Supplemental Logging Options Per Use Case

Logging option	GGSCI command	What it does	Use case
Table-level supplemental logging with unconditional logging for all supported columns. (See Enabling Table-level Supplemental Logging (page 3-7) for non-supported column types.)	`ADD TRANDATA` with `ALLCOLS` option	Enables unconditional supplemental logging of all of the columns of the table.	Used for bidirectional and active-active configurations where all column values are checked, not just the changed columns, when attempting to perform an update or delete. This takes more resources though allows for the highest level of real-time data validation and thus conflict detection. It can also be used when the source and target primary, unique, and foreign keys are not the same or are constantly changing between source and target.
Table-level supplemental logging, minimal setting	`ADD TRANDATA` with `NOSCHEDULINGCOLS` option	Enables unconditional supplemental logging of the primary key and all valid unique indexes of a table.	Use only for nonintegrated Replicat. This is the minimum required table-level logging.

- Enabling Minimum Database-level Supplemental Logging (page 3-4)
- Enabling Schema-level Supplemental Logging (page 3-5)
- Enabling Table-level Supplemental Logging (page 3-7)

3.2.1 Enabling Minimum Database-level Supplemental Logging

Oracle strongly recommends putting the Oracle source database into forced logging mode. Forced logging mode forces the logging of all transactions and loads, overriding any user or storage settings to the contrary. This ensures that no source data in the Extract configuration gets missed.

In addition, minimal supplemental logging, a database-level option, is required for an Oracle source database when using Oracle GoldenGate. This adds row chaining information, if any exists, to the redo log for update operations.

> **Note:**
>
> Database-level primary key (PK) and unique index (UI) logging is strongly discouraged because of the excessive additional overhead it creates on tables outside of replication. Unless those logging options are required for business purposes, you only need to enable minimal supplemental logging at the database level and force logging for Oracle GoldenGate.

Perform the following steps to verify and enable, if necessary, minimal supplemental logging and forced logging.

1. Log in to SQL*Plus as a user with `ALTER SYSTEM` privilege.
2. Issue the following command to determine whether the database is in supplemental logging mode and in forced logging mode. If the result is `YES` for both queries, the database meets the Oracle GoldenGate requirement.

   ```
   SELECT supplemental_log_data_min, force_logging FROM v$database;
   ```

3. If the result is `NO` for either or both properties, continue with these steps to enable them as needed:

   ```
   SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
   SQL> ALTER DATABASE FORCE LOGGING;
   ```

4. Issue the following command to verify that these properties are now enabled.

   ```
   SELECT supplemental_log_data_min, force_logging FROM v$database;
   ```

   The output of the query must be `YES` for both properties.

5. Switch the log files.

   ```
   SQL> ALTER SYSTEM SWITCH LOGFILE;
   ```

3.2.2 Enabling Schema-level Supplemental Logging

Oracle GoldenGate supports schema-level supplemental logging. Schema-level logging is required for an Oracle source database when using the Oracle GoldenGate DDL replication feature. In all other use cases, it is optional, but then you must use table-level logging instead (see Enabling Table-level Supplemental Logging (page 3-7)).

By default, schema-level logging automatically enables unconditional supplemental logging of the primary key and conditional supplemental logging of unique key(s) and foreign key(s) of all tables in a schema. Options enable you to alter the logging as needed.

> **Note:**
>
> Oracle strongly recommends using schema-level logging rather than table-level logging, because it ensures that any new tables added to a schema are captured if they satisfy wildcard specifications.

Perform the following steps on the source system to enable schema-level supplemental logging.

1. Apply Oracle Patch 13794550 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 with the alias of a user in the credential store who has privilege to enable schema-level supplemental logging.

   ```
   DBLOGIN USERIDALIAS alias
   ```

   See *Reference for Oracle GoldenGate for Windows and UNIX* for more information about `DBLOGIN` and additional options.
4. Issue the `ADD SCHEMATRANDATA` command for each schema for which you want to capture data changes with Oracle GoldenGate.

   ```
   ADD SCHEMATRANDATA schema [ALLCOLS | NOSCHEDULINGCOLS]
   ```

   Where:

   - Without options, `ADD SCHEMATRANDATA` schema enables the unconditional supplemental logging on the source system of the primary key and the conditional supplemental logging of all unique key(s) and foreign key(s) of all current and future tables in the given schema. Unconditional logging forces the primary key values to the log whether or not the key was changed in the current operation. Conditional logging logs all of the column values of a foreign or unique key if at least one of them was changed in the current operation. The default is optional to support nonintegrated Replicat but is required to support integrated Replicat because primary key, unique keys, and foreign keys must all be available to the inbound server to compute dependencies. For more information about integrated Replicat, see Deciding Which Apply Method to Use (page 5-5).
   - `ALLCOLS` can be used to enable the unconditional supplemental logging of all of the columns of a table and applies to all current and future tables in the given schema. Use to support integrated Replicat when the source and target tables have different scheduling columns. (*Scheduling columns* are the primary key, the unique key, and the foreign key.)
   - `NOSCHEDULINGCOLS` logs only the values of the primary key and all valid unique indexes for existing tables in the schema and new tables added later. This is the minimal required level of schema-level logging and is valid only for Replicat in nonintegrated mode.

   In the following example, the command enables default supplemental logging for the `finance` schema.

   ```
   ADD SCHEMATRANDATA finance
   ```

   In the following example, the command enables the supplemental logging only for the primary key and valid unique indexes for the `hr` schema.

   ```
   ADD SCHEMATRANDATA hr NOSCHEDULINGCOLS
   ```

   See *Reference for Oracle GoldenGate for Windows and UNIX* for more information about `ADD SCHEMATRANDATA`.

3.2.3 Enabling Table-level Supplemental Logging

Enable table-level supplemental logging on the source system in the following cases:

- To enable the required level of logging when not using schema-level logging (see Enabling Schema-level Supplemental Logging (page 3-5)). Either schema-level or table-level logging must be used. By default, table-level logging automatically enables unconditional supplemental logging of the primary key and conditional supplemental logging of unique key(s) and foreign key(s) of a table. Options enable you to alter the logging as needed.
- To prevent the logging of the primary key for any given table.
- To log non-key column values at the table level to support specific Oracle GoldenGate features, such as filtering and conflict detection and resolution logic.

Perform the following steps on the source system to enable table-level supplemental logging or use the optional features of the command.

1. Run GGSCI on the source system.
2. Issue the `DBLOGIN` command using the alias of a user in the credential store who has privilege to enable table-level supplemental logging.

   ```
   DBLOGIN USERIDALIAS alias
   ```

   See *Reference for Oracle GoldenGate for Windows and UNIX* for more information about `DBLOGIN` and additional options.
3. Issue the `ADD TRANDATA` command.

   ```
   ADD TRANDATA [container.]schema.table [, COLS (columns)] [, NOKEY] [, ALLCOLS |
   NOSCHEDULINGCOLS]
   ```

   Where:

   - *container* is the name of the root container or pluggable database if the table is in a multitenant container database.
   - *schema* is the source schema that contains the table.
   - *table* is the name of the table. See *Administering Oracle GoldenGate for Windows and UNIX* for instructions for specifying object names.
   - `ADD TRANDATA` without other options automatically enables unconditional supplemental logging of the primary key and conditional supplemental logging of unique key(s) and foreign key(s) of the table. Unconditional logging forces the primary key values to the log whether or not the key was changed in the current operation. Conditional logging logs all of the column values of a foreign or unique key if at least one of them was changed in the current operation. The default is optional to support nonintegrated Replicat (see also `NOSCHEDULINGCOLS`) but is required to support integrated Replicat because primary key, unique keys, and foreign keys must all be available to the inbound server to compute dependencies. For more information about integrated Replicat, see Deciding Which Apply Method to Use (page 5-5).
   - `ALLCOLS` enables the unconditional supplemental logging of all of the columns of the table. Use to support integrated Replicat when the source and target tables have different scheduling columns. (*Scheduling columns* are the primary key, the unique key, and the foreign key.)

- `NOSCHEDULINGCOLS` is valid for Replicat in nonintegrated mode only. It issues an `ALTER TABLE` command with an `ADD SUPPLEMENTAL LOG DATA ALWAYS` clause that is appropriate for the type of unique constraint that is defined for the table, or all columns in the absence of a unique constraint. This command satisfies the basic table-level logging requirements of Oracle GoldenGate when schema-level logging will not be used. See Ensuring Row Uniqueness in Source and Target Tables (page 10-1) for how Oracle GoldenGate selects a key or index.
- `COLS` *`columns`* logs non-key columns that are required for a `KEYCOLS` clause or for filtering and manipulation. The parentheses are required. These columns will be logged in addition to the primary key unless the `NOKEY` option is also present.
- `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 alternate `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.

See *Reference for Oracle GoldenGate for Windows and UNIX* for more information about `ADD TRANDATA`.

3.3 Enabling Oracle GoldenGate in the Database

The database services required to support Oracle GoldenGate capture and apply must be enabled explicitly for an Oracle 11.2.0.4 or greater database. This is required for all modes of Extract and Replicat.

To enable Oracle GoldenGate, set the following database initialization parameter. All instances in Oracle RAC must have the same setting.

```
ENABLE_GOLDENGATE_REPLICATION=true
```

For more information about this parameter, see Initialization Parameters.

3.4 Setting Flashback Query

To process certain update records, Extract fetches additional row data from the source database. Oracle GoldenGate fetches data for the following:

- User-defined types
- Nested tables
- XMLType objects

By default, Oracle GoldenGate uses Flashback Query to fetch the values from the undo (rollback) tablespaces. That way, Oracle GoldenGate can reconstruct a read-consistent row image as of a specific time or SCN to match the redo record.

For best fetch results, configure the source database as follows:

1. Set a sufficient amount of redo retention by setting the Oracle initialization parameters `UNDO_MANAGEMENT` and `UNDO_RETENTION` as follows (in seconds).

```
UNDO_MANAGEMENT=AUTO

UNDO_RETENTION=86400

UNDO_RETENTION can be adjusted upward in high-volume environments.
```

2. Calculate the space that is required in the undo tablespace by using the following formula.

   undo_space = UNDO_RETENTION * *UPS* + *overhead*

   Where:

   - *undo_space* is the number of undo blocks.
   - UNDO_RETENTION is the value of the UNDO_RETENTION parameter (in seconds).
   - *UPS* is the number of undo blocks for each second.
   - *overhead* is the minimal overhead for metadata (transaction tables, etc.).

   Use the system view V$UNDOSTAT to estimate *UPS* and *overhead*.

3. For tables that contain LOBs, do one of the following:

   - Set the LOB storage clause to RETENTION. This is the default for tables that are created when UNDO_MANAGEMENT is set to AUTO.
   - If using PCTVERSION instead of RETENTION, set PCTVERSION to an initial value of 25. You can adjust it based on the fetch statistics that are reported with the STATS EXTRACT command (see Table 3-2 (page 3-9)). If the value of the STAT_OPER_ROWFETCH CURRENTBYROWID or STAT_OPER_ROWFETCH_CURRENTBYKEY field in these statistics is high, increase PCTVERSION in increments of 10 until the statistics show low values.

4. Grant either of the following privileges to the Oracle GoldenGate Extract user:

   ```
   GRANT FLASHBACK ANY TABLE TO db_user

   GRANT FLASHBACK ON schema.table TO db_user
   ```

Oracle GoldenGate provides the following parameters to manage fetching.

Table 3-2 Oracle GoldenGate Parameters and Commands to Manage Fetching

Parameter or Command	Description
STATS EXTRACT command with REPORTFETCH option	Shows Extract fetch statistics on demand.
STATOPTIONS parameter with REPORTFETCH option	Sets the STATS EXTRACT command so that it always shows fetch statistics.
MAXFETCHSTATEMENTS parameter	Controls the number of open cursors for prepared queries that Extract maintains in the source database, and also for SQLEXEC operations.
MAXFETCHSTATEMENTS parameter	Controls the default fetch behavior of Extract: whether Extract performs a flashback query or fetches the current image from the table.
FETCHOPTIONS parameter with the USELATESTVERSION or NOUSELATESTVERSION option	Handles the failure of an Extract flashback query, such as if the undo retention expired or the structure of a table changed. Extract can fetch the current image from the table or ignore the failure.
REPFETCHEDCOLOPTIONS parameter	Controls the response by Replicat when it processes trail records that include fetched data or column-missing conditions.

3.5 Managing Server Resources

In integrated mode, Extract interacts with an underlying logmining server in the source database and Replicat interacts with an inbound server in the target database. This section provides guidelines for managing the shared memory consumed by the these servers.

The shared memory that is used by the servers comes from the Streams pool portion of the System Global Area (SGA) in the database. Therefore, you must set the database initialization parameter `STREAMS_POOL_SIZE` high enough to keep enough memory available for the number of Extract and Replicat processes that you expect to run in integrated mode. Note that Streams pool is also used by other components of the database (like Oracle Streams, Advanced Queuing, and Datapump export/import), so make certain to take them into account while sizing the Streams pool for Oracle GoldenGate.

By default, one integrated capture Extract requests the logmining server to run with `MAX_SGA_SIZE` of 1GB. Thus, if you are running three Extracts in integrated capture mode in the same database instance, you need at least 3 GB of memory allocated to the Streams pool. As a best practice, keep 25 percent of the Streams pool available. For example, if there are 3 Extracts in integrated capture mode, set `STREAMS_POOL_SIZE` for the database to the following value:

```
3 GB * 1.25 = 3.75 GB
```

4

Establishing Oracle GoldenGate Credentials

This chapter provides guidelines for creating database users for the processes that will interact with the database, assigning the correct privileges, and securing the credentials from unauthorized use.

Topics:

4.1 Assigning Credentials to Oracle GoldenGate

The Oracle GoldenGate processes require one or more database credentials with the correct database privileges for the database version, database configuration, and Oracle GoldenGate features that you are using. Create a source database user and a target database user, each one dedicated to Oracle GoldenGate on the source and target systems. The assigned user can be the same user for all of the Oracle GoldenGate processes that must connect to a source or target Oracle Database.

The following sections outline the Oracle GoldenGate processes that require user credentials:

Granting the Appropriate User Privileges (page 4-2) outlines the database privileges that each of those processes requires.

4.1.1 Extract User

The Extract user performs metadata queries on the source database and fetches data from the source tables when needed. In a local mining deployment of integrated capture, this user also creates, alters, and connects to the logmining server and receives logical change records (LCR) from it. (See Deciding Which Capture Method to Use (page 5-2) for more information about capture modes.)

If the source database is a multitenant container database, the Extract user must be a common user and must log into the root container. See Configuring Oracle GoldenGate in a Multitenant Container Database (page 6-1) for more information.

You need to assign an additional user if Extract will be operating in integrated capture mode and you are using a downstream mining database. This user will be the mining user and is created in the downstream database. The mining user creates, alters, and connects to the logmining server on the mining database, and it receives logical change records (LCR) from it. This user can be the same as the source Extract user or different. Choose the name of the mining user carefully. Once created by this user, the database logmining server cannot be altered or used by another user. See Configuring a Downstream Mining Database (page B-1) for more information about configuring downstream mining.

4.1.2 Replicat User

The Replicat user creates the Replicat checkpoint table (if used) and applies DML and DDL operations through Oracle Call Interface or through a database inbound server, depending on the Replicat mode. (See Deciding Which Apply Method to Use (page 5-5) for more information about Replicat modes.)

4.1.3 Other Oracle GoldenGate Users

A user is required in the source database for the Manager process if you are using Oracle GoldenGate DDL support. This user performs maintenance on the Oracle GoldenGate database objects that support DDL capture.

A user is required in either the source or target database for the `DEFGEN` utility. The location depends on where the data definition file is being generated. This user performs local metadata queries to build a data-definitions file that supplies the metadata to remote Oracle GoldenGate instances. For more information about the data-definitions file, see *Administering Oracle GoldenGate for Windows and UNIX*.

Additional users or privileges may be required to use the following features, if Extract will run in classic capture mode:

- RMAN log retention (see Log Retention Options (page 11-11))
- TDE support (see Configuring Oracle TDE Data in Classic Capture Mode (page 11-1))
- ASM (see Mining ASM-stored Logs in Classic Capture Mode (page 11-8))

4.1.4 Granting the Appropriate User Privileges

The user privileges that are required for Oracle GoldenGate depend on the database version and the Extract or Replicat process mode. For more information about process modes, see Choosing Capture and Apply Modes (page 5-1).

4.1.4.1 Oracle 11.2.0.4 or Later Database Privileges

The following privileges apply to Oracle versions 11.2.0.4 or later.

Table 4-1 Oracle GoldenGate Privileges, Version 11.2.0.4 or Later

Privilege	Extract Classic Mode	Extract Integrated Mode	Replicat All Modes	Purpose
`CREATE SESSION`	X	X	X	Connect to the database
`CONNECT`	X	X	X	For Replicat, required only if Replicat owns target objects. Alternatively, use `CREATE object`.
`RESOURCE`	X	X	X	Create objects If `RESOURCE` cannot be granted to Replicat, use: `ALTER USER user QUOTA {size | UNLIMITED} ON tablespace;`
`ALTER ANY TABLE`	X	X		Required for Oracle 12.1.0.1 only to issue the `ADD TRANDATA` command.
`ALTER SYSTEM`	X	X		Perform administrative changes, such as enabling logging
`ALTER USER <GGADMIN> set container_data=all container=current;`	X	X		Required for multitenant architecture and `<GGADMIN>` should be a valid Oracle GoldenGate administrator schema.
Privileges granted through `dbms_goldengate_auth.grant_admin_privilege`	X	X	X	(Extract) Grants privileges for both classic and integrated Extract, including the logmining server. (Replicat) Grants privileges for both nonintegrated and integrated replicat, including the database inbound server (Oracle 11.2.0.4 or later).
Any or all of optional privileges of `dbms_goldengate_auth.grant_admin_privilege`	X	X	X	• Capture from Data Vault • Capture from Virtual Private Database • Capture redacted data See Optional Grants for dbms_goldengate_auth.grant_admin_privilege (page 4-6) for more information.
`INSERT`, `UPDATE`, `DELETE` on target tables			X	Apply replicated DML to target objects
`CREATE TABLE`			X	Create a checkpoint table in target database
DDL privileges on target objects (if using DDL support)			X	Issue replicated DDL on target objects
`DBA`	X		X	DDL and sequence support
`LOCK ANY TABLE`			X	Lock target tables. Only required for initial load using direct bulk load to SQL*Loader.

Table 4-1 (Cont.) Oracle GoldenGate Privileges, Version 11.2.0.4 or Later

Privilege	Extract Classic Mode	Extract Integrated Mode	Replicat All Modes	Purpose
`SELECT ANY TRANSACTION`	X			Use a newer Oracle ASM API. See Mining ASM-stored Logs in Classic Capture Mode (page 11-8).

4.1.4.2 Oracle 11.2.0.3 or Earlier Database Privileges

The following privileges apply to Oracle versions 11.2.0.3 or earlier.

Table 4-2 Oracle GoldenGate Privileges, Oracle 11.2.0.3 or Earlier

Privilege	Extract Classic Mode	Extract Integrated Mode	Replicat	Manager	Purpose
`CREATE SESSION` and `ALTER SESSION`	X	X	X		Connect to the database
`ALTER SYSTEM`	X	X			Perform administrative changes, such as enabling logging
`RESOURCE`	X	X	X		Create objects If `RESOURCE` cannot be granted to Replicat, use: `ALTER USER` *`user`* `QUOTA {`*`size`*` | UNLIMITED} ON `*`tablespace`*;
`CONNECT`	X	X	X		For Replicat, required only if Replicat owns target objects. Alternatively, use `CREATE` *`object`*.
`SELECT ANY DICTIONARY`	X	X	X		Query data dictionary objects in the SYS schema
`FLASHBACK ANY TABLE` or `FLASHBACK ON` *`schema.table`*	X	X			Make flashback queries
`SELECT ANY TABLE` or `SELECT` on a *`schema.table`*	X	X	X		Perform queries on any table
`SELECT` on `dba_clusters`	X	X			

Table 4-2 (Cont.) Oracle GoldenGate Privileges, Oracle 11.2.0.3 or Earlier

Privilege	Extract Classic Mode	Extract Integrated Mode	Replicat	Manager	Purpose
`INSERT`, `UPDATE`, `DELETE` on target tables			X		Apply replicated DML to target objects
`CREATE TABLE`				X	Create a checkpoint table in target database
`EXECUTE` on `DBMS_FLASHBACK` package	X	X			Call `DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER`
DDL privileges on target objects (if using DDL support)			X		Issue replicated DDL on target objects
`GGS_GGSUSER_ROLE` (if using DDL support)	X	X			DML privileges on Oracle GoldenGate DDL objects. Role is created by user with SYSDBA privilege during installation of DDL objects.
`DELETE` on Oracle GoldenGate DDL objects				X	Use parameters that maintain Oracle GoldenGate DDL objects
`LOCK ANY TABLE`			X		Lock target tables. Only required for initial load using direct bulk load to SQL*Loader.
`SELECT ANY TRANSACTION`	X				Use a newer Oracle ASM API. See Mining ASM-stored Logs in Classic Capture Mode (page 11-8).
Privileges granted through `dbms_streams_auth.grant_admin_privilege`		X			Interact with database logmining server
`EXECUTE` on `dbms_logmnr_d` package		X			Issue the `REGISTER EXTRACT` command Required for Oracle version >= 11.1.0.5 and <= 11.2.0.1.
`SELECT FROM sys.logmnr_buildlog`		X			Issue the `REGISTER EXTRACT` command Required for Oracle version >= 11.1.0.5 and <= 11.2.0.1.

4.1.4.3 About the dbms_goldengate_auth.grant_admin_privilege Package

Most of the privileges that are needed for Extract and Replicat to operate in classic and integrated mode are granted through the `dbms_goldengate_auth.grant_admin_privilege` package.

- The following grants base privileges for Oracle 11.2.0.4 and later. The first example is the default, which grants to both capture and apply. The second shows how to explicitly grant to either capture or apply (in this case, capture).

```
grant_admin_privilege('ggadm')
grant_admin_privilege('ggadm','capture');
```

- The following grants base privileges for Oracle 11.2.0.3. The first example is the default, which grants to both capture and apply. The second shows how to explicitly grant to capture.

```
grant_admin_privilege('ggadm',grant_select_privileges=>true)
grant_admin_privilege('ggadm','capture',grant_select_privileges=>true)
```

4.1.4.4 Optional Grants for dbms_goldengate_auth.grant_admin_privilege

Additional grants can be added to `dbms_goldengate_auth.grant_admin_privilege` to support the optional features shown in Table 4-3 (page 4-6).

Table 4-3 Optional dbms_goldengate_auth.grant_admin_privilege Grants

Role/Privilege	Extract Classic Mode	Extract Integrated Mode	Replicat	Purpose
`DV_GOLDENGATE_ADMIN`	X	X	X	Capture or apply when Oracle Data Vault is being used. Grant the `DV_ACCTMGR` role to create user accounts in an Oracle Database Vault environment. Also, grant `DV_GOLDENGATE_REDO_ACCESS` if using classic capture with `TRANLOGOPTIONS DBLOGREADER` on Oracle Data Vault. See Mining ASM-stored Logs in Classic Capture Mode (page 11-8). Also, grant Replicat the privileges in `DBMS_MACADM.ADD_AUTH_TO_REALM` if applying to a realm.
`EXEMPT ACCESS POLICY`	X	X	X	Capture or apply when Oracle Virtual Private Database is being used.
`EXEMPT REDACTION POLICY`	X	X	X	Capture or apply redacted data

One or more of these privileges can be granted as a comma-separated list to 'CAPTURE', 'APPLY', or '*'. The default is * (capture and apply).

- In the following example, the Extract (capture) user ggadm is being granted DV_GOLDENGATE_ADMIN and EXEMPT REDACTION POLICY in addition to the privileges needed for Extract.

  ```
  dbms_goldengate_auth.grant_admin_privilege('ggadm','capture',
  grant_optional_privilege=>'DV_GOLDENGATE_ADMIN,EXEMPT REDACTION POLICY')
  ```

- In the following example, the Extract (capture) user ggadm is being granted all optional privileges in addition to the privileges needed for Extract:

  ```
  dbms_goldengate_auth.grant_admin_privilege('ggadm','capture',
  grant_optional_privileges=>'*')
  ```

- For Database Vault 12c environments with the Extract in integrated mode, the Database Vault owner needs to grant additionally the following two privileges:

  –
  ```
  BEGIN
    DVSYS.DBMS_MACADM.ADD_AUTH_TO_REALM
      (REALM_NAME    => 'Oracle Default Component Protection Realm'
      ,GRANTEE       => '<GGADMIN>'
      ,AUTH_OPTIONS => DBMS_MACUTL.G_REALM_AUTH_OWNER) ;
  END ;
  /
  ```

  –
  ```
  BEGIN
    dbms_macadm.authorize_ddl('SYS','SYSTEM');
  END;
  /
  ```

- For Oracle Transparent Data Encryption (TDE) environments with the Extract in classic mode, you have to additionally install the prvtclkm.plb package from the ORACLE_HOME/rdbms/admin and, grant the execute privilege to the Oracle GoldenGate administration user, GGADMIN:

  ```
  grant execute on sys.dbms_internal_clkm to GGADMIN
  ```

- If you want to create a spatial index in a different schema, the Oracle GoldenGate administration user, GGADMIN must have explicit CREATE TABLE, CREATE SEQUENCE, select, and index on the table privileges granted to the user:

  ```
  grant create table,create sequence to GGADMIN;
  ```

  This a one-time action, which could be done it you want to use in spatial replication. You could also use dbms_goldengate_auth.grant_admin_privilege as described in this section.

  The GGADMIN user must have select, and index on each table with an SDO column that an index will be created on:

  ```
  grant select, index on tklmob31.cola_markets1 to GGADMIN;
  ```

  The table must exist on the target, and the you have to manually issue this grant statement on the target, before issuing a Create index ... indextype is MDSYS.SPATIAL_INDEX on the source. The grant statement can't be issued on the source and replicated, because the applying the user GGADMIN can't grant itself privileges.

For more information about `dbms_goldengate_auth.grant_admin_privilege`, see *Oracle Database PL/SQL Packages and Types Reference*.

For more information about Data Vault requirements, see *Oracle Database Vault Administrator's Guide*.

4.2 Securing the Oracle GoldenGate Credentials

To preserve the security of your data, and to monitor Oracle GoldenGate processing accurately, do not permit other users, applications, or processes to log on as, or operate as, an Oracle GoldenGate database user.

Oracle GoldenGate provides different options for securing the login credentials assigned to Oracle GoldenGate processes. The recommended option is to use a credential store. You can create one credential store and store it in a shared location where all installations of Oracle GoldenGate can access it, or you can create a separate one on each system where Oracle GoldenGate is installed.

The credential store stores the user name and password for each of the assigned Oracle GoldenGate users. A user ID is associated with one or more aliases, and it is the alias that is supplied in commands and parameter files, not the actual user name or password. The credential file can be partitioned into domains, allowing a standard set of aliases to be used for the processes, while allowing the administrator on each system to manage credentials locally.

See *Administering Oracle GoldenGate for Windows and UNIX* for more information about creating a credential store and adding user credentials.

5

Choosing Capture and Apply Modes

This chapter contains information that helps you determine the appropriate capture and apply modes for your database environment.

Topics:

5.1 Overview of Oracle GoldenGate Capture and Apply Processes

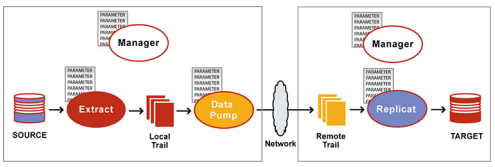
The Oracle GoldenGate capture process is known as *Extract*. Each instance of an Extract process is known as a *group*, which includes the process itself and the associated files that support it.

An additional Extract process, known as a *data pump*, is recommended to be used on the source system, so that captured data can be persisted locally to a series of files known as a *trail*. The data pump does not capture data but rather reads the local trail and propagates the data across the network to the target.

The Oracle GoldenGate apply process is known as *Replicat*. Each instance of a Replicat process is known as a *group*, which includes the process itself and the associated files that support it. Replicat reads data that is sent to local storage, known as a *trail*, and applies it to the target database.

Figure 5-1 (page 5-1) illustrates the basic Oracle GoldenGate process configuration. For more information about Oracle GoldenGate processes, see *Administering Oracle GoldenGate for Windows and UNIX*.

Figure 5-1 The Basic Configuration



> **Note:**
>
> Oracle Databases must be in `ARCHIVELOG` mode so that Extract can process the log files.

5.2 Deciding Which Capture Method to Use

For an Oracle source database, you can run Extract in either *classic capture* or *integrated capture* mode. The method that you use determines how you configure the Oracle GoldenGate processes and depends on such factors as:

- the data types involved
- the database configuration
- the version of the Oracle Database

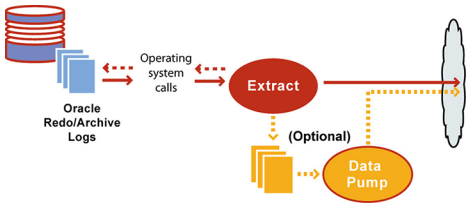
The following explains these modes and the database versions that each mode supports.

- About Classic Capture (page 5-2)
- About Integrated Capture (page 5-3)

5.2.1 About Classic Capture

In classic capture mode, the Oracle GoldenGate Extract process captures data changes from the Oracle redo or archive log files on the source system or from shipped archive logs on a standby system. Figure 5-2 (page 5-2) illustrates the configuration of Extract in classic capture mode.

Figure 5-2 Classic capture



Classic capture supports most Oracle data types fully, with restricted support for the complex data types. Classic capture is the original Oracle GoldenGate capture method. You can use classic capture for any source Oracle RDBMS that is supported by Oracle GoldenGate, with the exception of the multitenant container database.

You can use classic capture to support the following:

- UDTs, VARRAYs, `NOLOGGING` LOBs with source database compatibility set below 11.2.0.0.0.
- Transparent Data Encryption support with source database compatibility set below 11.0.0.0.0.

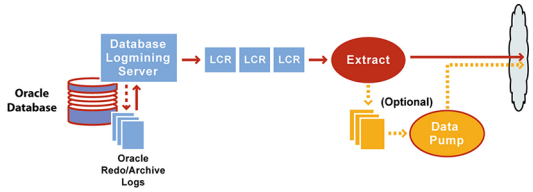
- `SECUREFILE` LOB support with source database compatibility set below 11.2.0.0.0.
- `NOLOGGING` LOB support with source database compatibility set below 11.2.0.0.0.

For more information, see Summary of Supported Oracle Data Types and Objects Per Capture Mode (page 1-7).

5.2.2 About Integrated Capture

In integrated capture mode, the Oracle GoldenGate Extract process interacts directly with a database logmining server to receive data changes in the form of logical change records (LCR). Figure 5-3 (page 5-3) illustrates the configuration of Extract in integrated capture mode.

Figure 5-3 Integrated Capture



Integrated capture supports more data and storage types as compared to classic capture, and the support is more transparent. For more information, see Summary of Supported Oracle Data Types and Objects Per Capture Mode (page 1-7).

The following are some additional benefits of integrated capture:

- Because integrated capture is fully integrated with the database, no additional setup is required to work with Oracle RAC, ASM, and TDE.
- Integrated capture uses the database logmining server to access the Oracle redo stream, with the benefit of being able to automatically switch between different copies of archive logs or different mirrored versions of the online logs. Thus integrated capture can transparently handle the absence of a log file caused by disk corruption, hardware failure, or operator error, assuming that additional copies of the archived and online logs are available
- Integrated capture enables faster filtering of tables.
- Integrated capture handles point-in-time recovery and RAC integration more efficiently.
- Integrated capture features integrated log management. The Oracle Recovery Manager (RMAN) automatically retains the archive logs that are needed by Extract.
- Integrated capture is the only mode that supports capture from a multitenant container database. One Extract can mine multiple pluggable databases within a multitenant container database.
- For a release 11.2.0.4 source database and later (with source compatibility set to 11.2.0.4 or higher), the capture of DDL is performed by the logmining server asynchronously and requires no special triggers, tables, or other database objects to be installed. Oracle GoldenGateupgrades can be performed without stopping user applications. The use of a DDL trigger and supporting objects is required

when Extract is in integrated mode with an Oracle 11*g* source database that is earlier than version 11.2.0.4.

- Because integrated capture and integrated apply are both database objects, the naming of the objects follow the same rules as other Oracle Database objects. For more information, see *Oracle Database Administrator's Guide*.
- Integrated Capture Supported Database Versions (page 5-4)
- Integrated Capture Deployment Options (page 5-4)

5.2.2.1 Integrated Capture Supported Database Versions

The database version determines the data type support available through integrated capture:

- **Full support**: To support all Oracle data and storage types, the compatibility setting of the source database must be at least 11.2.0.3 with the 11.2.0.3 Database specific bundle patch for Integrated Extract 11.2.x (My Oracle Support Document ID 1557031.1). To get this patch from My Oracle Support, go to:

  `https://support.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=1557031.1`

- To support DDL capture without installing DDL support objects, the source database must be at least Oracle 11.2.0.4 or later. For earlier database versions, DDL support objects are required. See Installing Trigger-Based DDL Capture (page D-1) for more information.
- **Limited support**: You can use integrated capture on an 11.2.0.3 downstream mining database for a source database with a compatibility less than 11.2.0.3, but in this mode, `SECUREFILE` LOBs, XML columns, Transparent Data Encryption, and UDTs have limited support based on database version and compatibility. The downstream mining database must have the 11.2.0.3 Database specific bundle patch for Integrated Extract 11.2.x (Doc ID 1557031.1) applied. See Integrated Capture Deployment Options (page 5-4). The downstream mining database must be at the same (or greater) database release (minimum 11.2.0.3) as the source database release being mined.

To understand the differences in data type support among different RDBMS versions, see Summary of Supported Oracle Data Types and Objects Per Capture Mode (page 1-7).

5.2.2.2 Integrated Capture Deployment Options

The deployment options for integrated capture are described in this section and depend on where the mining database is deployed. The mining database is the one where the logmining server is deployed.

- **Local deployment**: For a local deployment, the source database and the mining database are the same. The source database is the database for which you want to mine the redo stream to capture changes, and also where you deploy the logmining server. Because integrated capture is fully integrated with the database, this mode does not require any special database setup.
- **Downstream deployment**: In a downstream deployment, the source and mining databases are different databases. You create the logmining server at the downstream database. You configure redo transport at the source database to ship the redo logs to the downstream mining database for capture at that location.

Using a downstream mining server for capture may be desirable to offload the capture overhead and any other overhead from transformation or other processing from the production server, but requires log shipping and other configuration.

When using a downstream mining configuration, the source database and mining database must be of the same platform. For example, if the source database is running on Windows 64-bit, the downstream database must also be on a Windows 64-bit platform. See Configuring a Downstream Mining Database (page B-1) and Example Downstream Mining Configuration (page C-1) to configure a downstream mining database.

- **Downstream Oracle Active DatavGuard deployment:** You can fetch from an Oracle Active Data Guard (ADG) using the `FETCHUSERID` or `FETCHUSERIDALIAS` parameters to configure `userid/pwd@adg`.
- **Downstream sourceless Extract deployment:** In the Extract parameter file, replace the `USERID` parameter with `NOUSERID`. You must use `TRANLOGOPTIONS MININGUSER`. This deployment requires that the source database redo is compatible with Oracle GoldenGate releases greater than or equal to 11.2.0.4. Extract obtains all required information from the downstream mining database. Extract is not dependent on any connection to the source database. The source database can be shutdown and restarted without affecting Extract.

  Extract will abend if it encounters redo changes that require data to be fetched from the source database.
- **Integrated Extract from an ADG standby database deployment:** Traditionally, when Extract fetches data it is connected to the source database and fetches from it. Using this deployment, the fetch can be offloaded to an ADG standby database. This reduces the processing needed on the source database to enable replication. When used in conjunction with a downstream and sourceless configuration it almost completely eliminates any required processing by the source database.

5.3 Deciding Which Apply Method to Use

The Replicat process is responsible for the application of replicated data to an Oracle target database. For more information about Oracle GoldenGate processes, see *Administering Oracle GoldenGate for Windows and UNIX*.

For an Oracle target database, you can run Replicat in either *nonintegrated* or *integrated* mode. The following explains these modes and the database versions that each mode supports.

5.3.1 About Nonintegrated Replicat

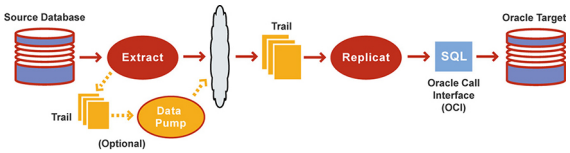
In nonintegrated mode, the Replicat process uses standard SQL to apply data directly to the target tables. In this mode, Replicat operates as follows:

- Reads the Oracle GoldenGate trail.
- Performs data filtering, mapping, and conversion.
- Constructs SQL statements that represent source database DML or DDL transactions (in committed order).

- Applies the SQL to the target through Oracle Call Interface (OCI).

Figure 5-4 (page 5-6) illustrates the configuration of Replicat in nonintegrated mode.

Figure 5-4 Nonintegrated Replicat



Use nonintegrated Replicat when:

- The target Oracle Database is a version earlier than Oracle 11.2.0.4.
- You want to make heavy use of features that are not supported in integrated Replicat mode. See About Integrated Replicat (page 5-6) for more information.

You can apply transactions in parallel with a nonintegrated Replicat by using a coordinated Replicat configuration. See *Administering Oracle GoldenGate for Windows and UNIX* for more information.

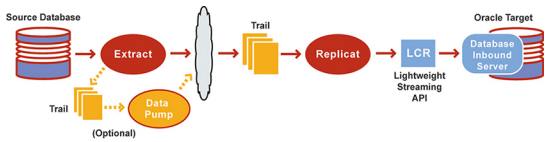
5.3.2 About Integrated Replicat

In integrated mode, the Replicat process leverages the apply processing functionality that is available within the Oracle Database. In this mode, Replicat operates as follows:

- Reads the Oracle GoldenGate trail.
- Performs data filtering, mapping, and conversion.
- Constructs logical change records (LCR) that represent source database DML transactions (in committed order). DDL is applied directly by Replicat.
- Attaches to a background process in the target database known as a *database inbound server* by means of a lightweight streaming interface.
- Transmits the LCRs to the inbound server, which applies the data to the target database.

Figure 5-5 (page 5-6) illustrates the configuration of Replicat in integrated mode.

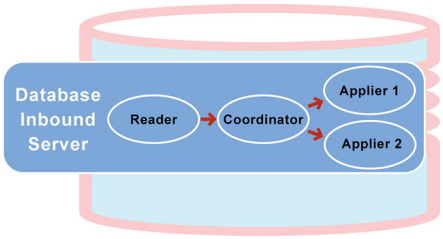
Figure 5-5 Integrated Replicat



Within a single Replicat configuration, multiple inbound server child processes known as *apply servers* apply transactions in parallel while preserving the original transaction atomicity. You can increase this parallelism as much as your target system will support

when you configure the Replicat process or dynamically as needed. Figure 5-6 (page 5-7) illustrates integrated Replicat configured with two parallel apply servers.

Figure 5-6 Integrated Replicat with Two Parallel Apply Servers



Integrated Replicat applies transactions asynchronously. Transactions that do not have interdependencies can be safely executed and committed out of order to achieve fast throughput. Transactions with dependencies are guaranteed to be applied in the same order as on the source.

A reader process in the inbound server computes the dependencies among the transactions in the workload based on the constraints defined at the target database (primary key, unique, foreign key). Barrier transactions and DDL operations are managed automatically, as well. A coordinator process coordinates multiple transactions and maintains order among the apply servers.

If the inbound server does not support a configured feature or column type, Replicat disengages from the inbound server, waits for the inbound server to complete transactions in its queue, and then applies the transaction to the database in *direct apply* mode through OCI. Replicat resumes processing in integrated mode after applying the direct transaction.

The following features are applied in direct mode by Replicat:

- DDL operations
- Sequence operations
- `SQLEXEC` parameter within a `TABLE` or `MAP` parameter
- `EVENTACTIONS` processing
- UDT Note, if the extract uses `USENATIVEOBJSUPPORT` to capture the UDT, then integrated Replicat will apply it with the inbound server, otherwise it will be handled by Replicat directly.

Because transactions are applied serially in direct apply mode, heavy use of such operations may reduce the performance of the integrated Replicat mode. Integrated Replicat performs best when most of the apply processing can be performed in integrated mode. See Monitoring and Controlling Processing After the Instantiation (page 15-13).

> **Note:**
>
> User exits are executed in integrated mode. The user exit may produce unexpected results, however, if the exit code depends on data in the replication stream.

5.3.2.1 Benefits of Integrated Replicat

The following are the benefits of using integrated Replicat versus nonintegrated Replicat.

- Integrated Replicat enables heavy workloads to be partitioned automatically among parallel apply processes that apply multiple transactions concurrently, while preserving the integrity and atomicity of the source transaction. Both a minimum and maximum number of apply processes can be configured with the `PARALLELISM` and `MAX_PARALLELISM` parameters. Replicat automatically adds additional servers when the workload increases, and then adjusts downward again when the workload lightens.
- Integrated Replicat requires minimal work to configure. All work is configured within one Replicat parameter file, without configuring range partitions.
- High-performance apply streaming is enabled for integrated Replicat by means of a lightweight application programming interface (API) between Replicat and the inbound server.
- Barrier transactions are coordinated by integrated Replicat among multiple server apply processes.
- DDL operations are processed as direct transactions that force a barrier by waiting for server processing to complete before the DDL execution.
- Transient duplicate primary key updates are handled by integrated Replicat in a seamless manner.
- Integrated Replicat works with single or pluggable databases.

5.3.2.2 Integrated Replicat Requirements

To use integrated Replicat, the following must be true.

- The target Oracle Database must be Oracle 11.2.0.4 or later.
- Supplemental logging must be enabled on the source database to support the computation of dependencies among tables and scheduling of concurrent transactions on the target. Instructions for enabling the required logging are in Configuring Logging Properties (page 3-2). This logging can be enabled at any time up to, but before, you start the Oracle GoldenGate processes.

5.4 Using Different Capture and Apply Modes Together

You can use the following capture and apply modes together:

- Classic capture (Oracle or non-Oracle source) and nonintegrated Replicat
- Classic capture (Oracle or non-Oracle source) and integrated Replicat
- Integrated capture and nonintegrated Replicat
- Integrated capture and integrated Replicat

You can use integrated capture and classic capture concurrently within the same source Oracle GoldenGate instance, and you can use integrated Replicat and

nonintegrated Replicat concurrently within the same target Oracle GoldenGate instance. This configuration requires careful placement of your objects within the appropriate process group, because there is no coordination of DDL or DML between classic and integrated capture modes, nor between nonintegrated and integrated Replicat modes. Each Extract group must process objects that are suited to the processing mode, based on table data types and attributes. No objects in one Extract can have DML or DDL dependencies on objects in the other Extract. The same type of segregation must be applied to the Replicat configuration.

The recommended Oracle GoldenGate configuration, when supported by the Oracle version, is to use one integrated capture on an Oracle source and one integrated Replicat per source database on an Oracle target. Integrated capture supports certain data types more completely than classic capture. One integrated Replicat configuration supports all Oracle data types either through the inbound server or by switching to direct apply when necessary, and it preserves source transaction integrity. You can adjust the parallelism settings to the desired apply performance level as needed.

If the target database is an Oracle version that does not support integrated Replicat, or if it is a non-Oracle Database, you can use a coordinated Replicat configuration. See *Administering Oracle GoldenGate for Windows and UNIX* for more information.

5.5 Switching to a Different Process Mode

You can switch between process modes. For example, you can switch from classic capture to integrated capture, or from integrated capture to classic capture. For instructions, see *Administering Oracle GoldenGate for Windows and UNIX*.

6

Configuring Oracle GoldenGate in a Multitenant Container Database

This chapter contains additional configuration instructions when configuring Oracle GoldenGate in a multitenant container database (CDB).

Topics:

6.1 Using Oracle GoldenGate with Pluggable Databases

In most ways, Oracle GoldenGate operates in a multitenant container database the same way that it operates in a regular Oracle Database. Therefore, consider these instructions as the foundation for following the actual configuration instructions in the following chapters:

Configuring Capture in Integrated Mode (page 7-1)

Configuring Oracle GoldenGate Apply (page 9-1)

6.1.1 Capturing from Pluggable Databases

One Extract group can capture from multiple pluggable databases to a single trail. In the parameter file, source objects must be specified in `TABLE` and `SEQUENCE` statements with their fully qualified three-part names in the format of *`container.schema.object`*.

As an alternative to specifying three-part names, you can specify a default pluggable database with the `SOURCECATALOG` parameter, and then specify only the *`schema.object`* in subsequent `TABLE` or `SEQUENCE` parameters. You can use multiple instances of this configuration to handle multiple source pluggable databases. For example:

```
SOURCECATALOG pdb1
TABLE schema*.tab*;
SEQUENCE schema*.seq*;
SOURCECATALOG pdb2
TABLE schema*.tab*;
SEQUENCE schema*.seq*;
```

6.1.2 Applying to Pluggable Databases

Replicat can only connect and apply to one pluggable database. To specify the correct one, use a SQL*Net connect string for the database user that you specify with the `USERID` or `USERIDALIAS` parameter. For example: `GGADMIN@FINANCE`. In the parameter file,

specify only the *schema.object* in the TARGET portion of the MAP statements. In the MAP portion, identify source objects captured from more than one pluggable database with their three-part names or use the SOURCECATALOG parameter with two-part names. The following is an example of this configuration.

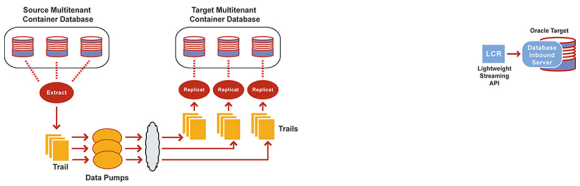
```
SOURCECATALOG pdb1
MAP schema*.tab*, TARGET *1.*;
MAP schema*.seq*, TARGET *1.*;
SOURCECATALOG pdb2
MAP schema*.tab*, TARGET *2.*;
MAP schema*.seq*, TARGET *2.*;
```

The following is an example *without* the use of SOURCECATALOG to identify the source pluggable database. In this case, the source objects are specified with their three-part names.

```
MAP pdb1.schema*.tab*, TARGET *1.*;
MAP pdb1.schema*.seq*, TARGET *1.*;
```

To configure replication from multiple source pluggable databases to multiple target pluggable databases, you can configure parallel Extract and Replicat streams, each handling data for one pluggable database. Alternatively, you can configure one Extract capturing from multiple source pluggable databases, which writes to one trail that is read by multiple Replicat groups, each applying to a different target pluggable database. Yet another alternative is to use one Extract writing to multiple trails, each trail read by a Replicat assigned to a specific target pluggable database (see Figure 6-1 (page 6-2)).

Figure 6-1 Possible Configuration for Capture and Apply from Multiple Pluggable Databases



6.1.3 Excluding Objects from the Configuration

To exclude pluggable databases, schemas, and objects from the configuration, you can use the CATALOGEXCLUDE, SCHEMAEXCLUDE, TABLEEXCLUDE, MAPEXCLUDE, and EXCLUDEWILDCARDOBJECTSONLY parameters. See *Reference for Oracle GoldenGate for Windows and UNIX* for more information.

6.2 Other Requirements for Multitenant Container Databases

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

- The different pluggable databases in the multitenant container database can have different character sets. Oracle GoldenGate captures data from any multitenant

database with different character sets into one trail file and replicates the data without corruption due to using different character sets.

- Extract must operate in integrated capture mode. See Deciding Which Capture Method to Use (page 5-2) for more information about Extract capture modes. Replicat can operate in any of its modes.
- Extract must connect to the root container (`cdb$root`) as a common user in order to interact with the logmining server. To specify the root container, use the appropriate SQL*Net connect string for the database user that you specify with the `USERID` or `USERIDALIAS` parameter. For example: `C##GGADMIN@FINANCE`. See Establishing Oracle GoldenGate Credentials (page 4-1) for how to create a user for the Oracle GoldenGate processes and grant the correct privileges.
- To support source CDB 12.2, Extract must specify the trail format as release 12.3.
- The `dbms_goldengate_auth.grant_admin_privilege` package grants the appropriate privileges for capture and apply within a multitenant container database. This includes the `container` parameter, which must be set to `ALL`, as shown in the following example:

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

See the following chapters to begin configuring the Oracle GoldenGate processes:

Configuring Capture in Integrated Mode (page 7-1)

Configuring Oracle GoldenGate Apply (page 9-1)

7

Configuring Capture in Integrated Mode

This chapter contains instructions for configuring the Oracle GoldenGate capture process to capture transaction data in integrated mode. See Choosing Capture and Apply Modes (page 5-1) for more information about Extract modes.

> **Note:**
>
> To switch an active Extract configuration from classic to integrated mode, perform these configuration steps and then see *Administering Oracle GoldenGate for Windows and UNIX*.

Topics:

7.1 Prerequisites for Configuring Integrated Capture

The guidelines in the following sections should be satisfied before configuring Extract in integrated mode.

1. Preparing the Database for Oracle GoldenGate (page 3-1)
2. Establishing Oracle GoldenGate Credentials (page 4-1)
3. Choosing Capture and Apply Modes (page 5-1)
4. Create the Oracle GoldenGate instance on the source system by configuring the Manager process. See *Administering Oracle GoldenGate for Windows and UNIX*.
5. Additionally, review the guidelines in *Administering Oracle GoldenGate for Windows and UNIX*

7.2 What to Expect from these Instructions

These instructions show you how to configure a basic Extract parameter (configuration) file for the primary Extract, which captures transaction data from the data source, and for a data-pump Extract, which propagates captured data that is stored locally in a *trail* from the source system to the target system. Your business requirements probably will require a more complex topology, but this procedure forms a basis for the rest of your configuration steps.

By performing these steps, you can:

- get the basic configuration files established.
- build upon them later by adding more parameters as you make decisions about features or requirements that apply to your environment.
- use copies of them to make the creation of additional parameter files faster than starting from scratch.

7.3 Configuring the Primary Extract in Integrated Capture Mode

The mining database, from which the primary Extract captures log change records from the logmining server, can be either local or downstream from the source database. These steps configure the primary Extract to capture transaction data in integrated mode from either location. See Configuring a Downstream Mining Database (page B-1) and Example Downstream Mining Configuration (page C-1) for more information about capturing from a downstream mining database.

> **Note:**
>
> One Extract group is generally sufficient to capture from a single database or multiple pluggable databases within a multitenant container database. See Configuring Oracle GoldenGate in a Multitenant Container Database (page 6-1).

1. In GGSCI on the source system, create the Extract parameter file.

   ```
   EDIT PARAMS name
   ```

   Where: *name* is the name of the primary Extract.

2. Enter the Extract parameters in the order shown, starting a new line for each parameter statement. Examples are shown for a regular database and a multitenant container database. The difference between the two is whether you must use two-part or three-part object names in the `TABLE` and `SEQUENCE` specifications. See Table 7-1 (page 7-4) for more information and parameter descriptions.

   Basic parameters for Extract where mining database is a downstream database and is a regular database. and is a regular database with source ADG used for FETCH

   ```
   EXTRACT financep
   USERIDALIAS tiger1
   LOGALLSUPCOLS
   UPDATERECORDFORMAT COMPACT
   DDL INCLUDE MAPPED
   ENCRYPTTRAIL AES192
   EXTTRAIL /ggs/dirdat/lt
   SEQUENCE hr.employees_seq;
   TABLE hr.*;
   ```

Basic parameters for Extract where mining database is a downstream database and is a multitenant container database with source ADG used for FETCH

```
EXTRACT financep
USERIDALIAS tiger1
LOGALLSUPCOLS
UPDATERECORDFORMAT COMPACT
DDL INCLUDE MAPPED SOURCECATALOG pdb1 INCLUDE MAPPED SOURCECATALOG pdb2
ENCRYPTTRAIL AES192
EXTTRAIL /ggs/dirdat/lt
TABLE test.ogg.tab1;
SOURCECATALOG pdb1
SEQUENCE hr.employees_seq;
TABLE hr.*;
SOURCECATALOG pdb2
TABLE sales.*;
TABLE acct.*;
```

Basic parameters for Extract where the mining database is a downstream database and is a regular database

```
EXTRACT financep
USERIDALIAS tiger1
TRANLOGOPTIONS MININGUSERALIAS tiger2
TRANLOGOPTIONS INTEGRATEDPARAMS (MAX_SGA_SIZE 164, &
   DOWNSTREAM_REAL_TIME_MINE y)
LOGALLSUPCOLS
UPDATERECORDFORMAT COMPACT
DDL INCLUDE MAPPED
ENCRYPTTRAIL AES192
EXTTRAIL /ggs/dirdat/lt
SEQUENCE hr.employees_seq;
TABLE hr.*;
```

Basic parameters for the primary Extract where the mining database is a downstream database and is a multitenant container database

```
EXTRACT financep
USERIDALIAS tiger1
TRANLOGOPTIONS MININGUSERALIAS tiger2
TRANLOGOPTIONS INTEGRATEDPARAMS (MAX_SGA_SIZE 164, &
   DOWNSTREAM_REAL_TIME_MINE y)
LOGALLSUPCOLS
UPDATERECORDFORMAT COMPACT
DDL INCLUDE MAPPED SOURCECATALOG pdb1 INCLUDE MAPPED SOURCECATALOG pdb2
ENCRYPTTRAIL AES192EXTTRAIL /ggs/dirdat/lt
TABLE test.ogg.tab1;
SOURCECATALOG pdb1
SEQUENCE hr.employees_seq;
TABLE hr.*;
SOURCECATALOG pdb2
TABLE sales.*;
TABLE acct.*;
```

Table 7-1 Basic Parameters for Primary Extract in Integrated Capture Mode

Parameter	Description
`EXTRACT group`	*group* is the name of the Extract group. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.
`USERIDALIAS alias`	Specifies the alias of the database login credential of the user that is assigned to Extract. This credential must exist in the Oracle GoldenGate credential store. For more information, see Establishing Oracle GoldenGate Credentials (page 4-1).
`LOGALLSUPCOLS`	Writes all supplementally logged columns to the trail, including those required for conflict detection and resolution and the scheduling columns required to support integrated Replicat. (Scheduling columns are primary key, unique index, and foreign key columns.) You configure the database to log these columns with GGSCI commands. See Configuring Logging Properties (page 3-2).
`UPDATERECORDFORMAT COMPACT`	Combines the before and after images of an `UPDATE` operation into a single record in the trail. This parameter is valid for Oracle Databases version 12c and later to support Replicat in integrated mode. Although not a required parameter, `UPDATERECORDFORMAT COMPACT` is a best practice and significantly improves Replicat performance. See *Reference for Oracle GoldenGate for Windows and UNIX* for more information.
`TRANLOGOPTIONS MININGUSERALIAS alias`	Specifies connection information for the logmining server at the downstream mining database, if being used. `MININGUSERALIAS` specifies the alias of the Extract user for the downstream mining database. This is the user that you created in Configuring a Downstream Mining Database (page B-1). The credential for this user must be stored in the Oracle GoldenGate credential store. See *Administering Oracle GoldenGate for Windows and UNIX* for more information. Use `MININGUSERALIAS` only if the database logmining server is in a different database from the source database; otherwise just use `USERIDALIAS`. When using `MININGUSERALIAS`, use it in addition to `USERIDALIAS`, because credentials are required for both databases.
`TRANLOGOPTIONS [INTEGRATEDPARAMS (parameter[, ...])]`	Optional, passes parameters to the Oracle Database that contains the database logmining server. Use only to change logmining server parameters from their default settings. See Additional Parameter Options for Integrated Capture (page A-1).
`TRANLOGOPTIONS CHECKPOINTRETENTIONTIME days`	Optional, controls the number of days that Extract retains checkpoints before purging them automatically. Partial days can be specified using decimal values. For example, 8.25 specifies 8 days and 6 hours. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.
`DDL include_clause`	Required if replicating DDL operations. See Configuring DDL Support (page 13-1) for more information.
`ENCRYPTTRAIL algorithm`	Encrypts the local trail. For more information about Oracle GoldenGate trail encryption options, see *Administering Oracle GoldenGate for Windows and UNIX*.
`EXTTRAIL pathname`	Specifies the path name of the local trail to which the primary Extract writes captured data. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*

Table 7-1 (Cont.) Basic Parameters for Primary Extract in Integrated Capture Mode

Parameter	Description
SOURCECATALOG *container*	Use this parameter when the source database is a multitenant container database. Specifies the name of a pluggable database that is to be used as the default container for all subsequent TABLE and SEQUENCE parameters that contain two-part names. This parameter enables you to use two-part object names (*schema.object*) rather than three-part names (*container.schema.object*). It remains in effect until another SOURCECATALOG parameter is encountered or a full three-part TABLE or SEQUENCE specification is encountered.
{TABLE | SEQUENCE} *[container.]schema.object*;	Specifies the database object for which to capture data. • TABLE specifies a table or a wildcarded set of tables. • SEQUENCE specifies a sequence or a wildcarded set of sequences. • *container* is the name of the pluggable database (PDB) that contains the object, if this database is a multitenant container database. The container part of the name is not required if this Extract group will only process data from one PDB and the default PDB is specified with the SOURCECATALOG parameter. See Configuring Oracle GoldenGate in a Multitenant Container Database (page 6-1) for additional configuration requirements. • *schema* is the schema name or a wildcarded set of schemas. • *object* is the table or sequence name, or a wildcarded set of those objects. See *Administering Oracle GoldenGate for Windows and UNIX* for information about how to specify object names with and without wildcards. Terminate the parameter statement with a semi-colon. To exclude a name from a wildcard specification, use the CATALOGEXCLUDE, SCHEMAEXCLUDE, TABLEEXCLUDE, and EXCLUDEWILDCARDOBJECTSONLY parameters as appropriate. For more information and for additional TABLE options that control data filtering, mapping, and manipulation, see *Reference for Oracle GoldenGate for Windows and UNIX*.

MAPINVISIBLECOLUMNS	Controls whether or not Replicat includes invisible columns in Oracle target tables for default column mapping. Configure the invisible columns in your column mapping using SQL to explicitly specify column names. For example:
	`CREATE TABLE tab1 (id NUMBER, data CLOB INVISIBLE);`
	`INSERT INTO tab1 VALUES (1, 'a');ERROR: ORA-913`
	`INSERT INTO tab1 (id, data) VALUES (1, 'a'); OK`
	You can change the column visibility using ALTER TABLE. The invisible column can be part of an index, including primary key and unique index.

3. Enter any optional Extract parameters that are recommended for your configuration. You can edit this file at any point before starting processing by using the EDIT PARAMS command in GGSCI. For more information, see the *Reference for Oracle GoldenGate for Windows and UNIX*.
4. Save and close the file.

7.4 Configuring the Data Pump Extract

These steps configure the data pump that reads the local trail and sends the data across the network to a remote trail. The data pump is optional, but recommended. For

more information about data pumps, see *Administering Oracle GoldenGate for Windows and UNIX*.

1. In GGSCI on the source system, create the data-pump parameter file.

   ```
   EDIT PARAMS name
   ```

   Where: *name* is the name of the data pump Extract.

2. Enter the data-pump parameters in the order shown, starting a new line for each parameter statement. Your input variables will be different. See Table 7-2 (page 7-6) for descriptions.

   Basic parameters for the data-pump Extract group using two-part object names:

   ```
   EXTRACT extpump
   USERIDALIAS tiger1
   RMTHOST fin1, MGRPORT 7809 ENCRYPT AES192, KEYNAME securekey2
   RMTTRAIL /ggs/dirdat/rt
   SEQUENCE hr.employees_seq;
   TABLE hr.*;
   ```

   Basic parameters for the data-pump Extract group using three-part object names (including a pluggable database):

   ```
   EXTRACT extpump
   USERIDALIAS tiger1
   RMTHOST fin1, MGRPORT 7809 ENCRYPT AES192, KEYNAME securekey2
   RMTTRAIL /ggs/dirdat/rt
   TABLE test.ogg.tab1;
   SOURCECATALOG pdb1
   SEQUENCE hr.employees_seq;
   TABLE hr.*;
   SOURCECATALOG pdb2
   TABLE sales.*;
   TABLE acct.*;
   ```

Table 7-2 Basic Parameters for a Data-pump Extract

Parameter	Description
`EXTRACT group`	*group* is the name of the data pump Extract. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.
`USERIDALIAS alias`	Specifies the alias of the database login credential of the user that is assigned to Extract. This credential must exist in the Oracle GoldenGate credential store. For more information, see Establishing Oracle GoldenGate Credentials (page 4-1)
`RMTHOST hostname, MGRPORT portnumber, [, ENCRYPT algorithm KEYNAME keyname]`	• `RMTHOST` specifies the name or IP address of the target system. • `MGRPORT` specifies the port number where Manager is running on the target. • `ENCRYPT` specifies optional encryption of data across TCP/IP. For additional options and encryption details, see *Reference for Oracle GoldenGate for Windows and UNIX*.
`RMTTRAIL pathname`	Specifies the path name of the remote trail. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.

Table 7-2 (Cont.) Basic Parameters for a Data-pump Extract

Parameter	Description
`SOURCECATALOG container`	Use this parameter when the source database is a multitenant container database. Specifies the name of a pluggable database that is to be used as the default container for all subsequent `TABLE` and `SEQUENCE` parameters that contain two-part names. This parameter enables you to use two-part object names (`schema.object`) rather than three-part names (`container.schema.object`). It remains in effect until another `SOURCECATALOG` parameter is encountered or a full three-part `TABLE` or `SEQUENCE` specification is encountered. Use this parameter when the source database is a multitenant container database. See *Reference for Oracle GoldenGate for Windows and UNIX* for more information about `SOURCECATALOG`.
`{TABLE | SEQUENCE} [container.]schema.object;`	Specifies a table or sequence, or multiple objects specified with a wildcard. In most cases, this listing will be the same as that in the primary Extract parameter file. • `TABLE` specifies a table or a wildcarded set of tables. • `SEQUENCE` specifies a sequence or a wildcarded set of sequences. • *`container`* is the name of the root container or pluggable database that contains the table or sequence, if this source database is a multitenant container database. See the `SOURCECATALOG` description in this table. • *`schema`* is the schema name or a wildcarded set of schemas. • *`object`* is the name of a table or sequence, or a wildcarded set of those objects. See *Administering Oracle GoldenGate for Windows and UNIX* for information about how to specify object names with and without wildcards. Terminate this parameter statement with a semi-colon. To exclude tables or sequences from a wildcard specification, use the `TABLEEXCLUDE` or `SEQUENCEEXCLUDE` parameter after the `TABLE` statement. For more information and for additional `TABLE` options that control data filtering, mapping, and manipulation, see *Reference for Oracle GoldenGate for Windows and UNIX*.

3. Enter any optional Extract parameters that are recommended for your configuration. You can edit this file at any point before starting processing by using the `EDIT PARAMS` command in GGSCI. For more information, see the *Reference for Oracle GoldenGate for Windows and UNIX*.
4. See Optional Parameters for Integrated Modes (page A-1) for additional configuration considerations.
5. Save and close the file.

7.5 Next Steps

Once you have created a basic parameter file for integrated capture, see the following for additional configuration steps.

Configuring Oracle GoldenGate Apply (page 9-1)

Configuring Oracle GoldenGate in a Multitenant Container Database (page 6-1)

Additional Oracle GoldenGate Configuration Considerations (page 10-1)

Configuring DDL Support (page 13-1) (to use Oracle GoldenGate DDL support)

Creating Process Groups (page 14-1)

Instantiating Oracle GoldenGate Replication (page 15-1)

8
Configuring Capture in Classic Mode

This chapter contains instructions for configuring the Oracle GoldenGate capture process in classic mode.

> **Note:**
>
> To switch an active Extract configuration from integrated to classic mode, perform these configuration steps and then see *Administering Oracle GoldenGate for Windows and UNIX*.

Topics:

8.1 Prerequisites for Configuring Classic Capture

The guidelines in the following sections should be satisfied before configuring Extract in integrated mode.

1. Preparing the Database for Oracle GoldenGate (page 3-1)
2. Establishing Oracle GoldenGate Credentials (page 4-1)
3. Choosing Capture and Apply Modes (page 5-1)
4. Create the Oracle GoldenGate instance on the source system by configuring the Manager process. See *Administering Oracle GoldenGate for Windows and UNIX*.
5. Additionally, review the guidelines in *Administering Oracle GoldenGate for Windows and UNIX*.

8.2 What to Expect from these Instructions

These instructions show you how to configure a basic Extract parameter (configuration) file for the primary Extract, which captures transaction data from the data source, and for a data-pump Extract, which propagates captured data that is stored locally in a `trail` from the source system to the target system. Your business requirements probably will require a more complex topology, but this procedure forms a basis for the rest of your configuration steps.

By performing these steps, you can:

- get the basic configuration files established.
- build upon them later by adding more parameters as you make decisions about features or requirements that apply to your environment.
- use copies of them to make the creation of additional parameter files faster than starting from scratch.

> **Note:**
>
> These instructions do not configure Oracle GoldenGate to perform DDL capture or replication. To support DDL, create the parameter files and then see the following chapters:
>
> Installing Trigger-Based DDL Capture (page D-1)
>
> Configuring DDL Support (page 13-1)

8.3 Configuring the Primary Extract in Classic Capture Mode

These steps configure Extract to capture transaction data in classic mode. See Choosing Capture and Apply Modes (page 5-1) for more information about Extract modes.

1. In GGSCI on the source system, create the Extract parameter file.

   ```
   EDIT PARAMS name
   ```

   Where: *name* is the name of the primary Extract.

2. Enter the Extract parameters in the order shown, starting a new line for each parameter statement. See Table 8-1 (page 8-2) for more information and parameter descriptions.

   Basic parameters for the primary Extract in classic capture mode

   ```
   EXTRACT finance
   USERIDALIAS tiger1
   LOGALLSUPCOLS
   ENCRYPTTRAIL AES192
   EXTTRAIL /ggs/dirdat/lt
   SEQUENCE hr.employees_seq;
   TABLE hr.*;
   ```

Table 8-1 Basic Parameters for Primary Extract in Classic Capture Mode

Parameter	Description
`EXTRACT group`	*group* is the name of the Extract group. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.
`USERIDALIAS alias`	Specifies the alias of the database login credential of the user that is assigned to Extract. This credential must exist in the Oracle GoldenGate credential store. For more information, see Establishing Oracle GoldenGate Credentials (page 4-1).

Table 8-1 (Cont.) Basic Parameters for Primary Extract in Classic Capture Mode

Parameter	Description
`LOGALLSUPCOLS`	Writes all supplementally logged columns to the trail, including those required for conflict detection and resolution and the scheduling columns required to support integrated Replicat. (Scheduling columns are primary key, unique index, and foreign key columns.) You configure the database to log these columns with GGSCI commands. See Configuring Logging Properties (page 3-2).
`UPDATERECORDFORMAT COMPACT`	Combines the before and after images of an `UPDATE` operation into a single record in the trail. This parameter is valid for Oracle Databases version 12c and later to support Replicat in integrated mode. Although not a required parameter, `UPDATERECORDFORMAT COMPACT` is a best practice and significantly improves Replicat performance. See *Reference for Oracle GoldenGate for Windows and UNIX* for more information.
`ENCRYPTTRAIL` *`algorithm`*	Encrypts the local trail. For more information about Oracle GoldenGate trail encryption options, see *Administering Oracle GoldenGate for Windows and UNIX*.
`EXTTRAIL` *`pathname`*	Specifies the path name of the local trail to which the primary Extract writes captured data. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*
`{TABLE | SEQUENCE}` *`schema.object`*`;`	Specifies the database object for which to capture data. • `TABLE` specifies a table or a wildcarded set of tables. • `SEQUENCE` specifies a sequence or a wildcarded set of sequences. • *`schema`* is the schema name or a wildcarded set of schemas. • *`object`* is the table or sequence name, or a wildcarded set of those objects. See *Administering Oracle GoldenGate for Windows and UNIX* for information about how to specify object names with and without wildcards. Terminate the parameter statement with a semi-colon. To exclude tables from a wildcard specification, use the `TABLEEXCLUDE` parameter. See *Reference for Oracle GoldenGate for Windows and UNIX* for more information about usage and syntax. For more information and for additional `TABLE` options that control data filtering, mapping, and manipulation, see *Reference for Oracle GoldenGate for Windows and UNIX*.

3. Enter any optional Extract parameters that are recommended for your configuration. You can edit this file at any point before starting processing by using the `EDIT PARAMS` command in GGSCI. For more information, see the *Reference for Oracle GoldenGate for Windows and UNIX*.
4. Save and close the file.

8.4 Configuring the Data Pump Extract

These steps configure the data pump that reads the local trail and sends the data across the network to a remote trail. The data pump is optional, but recommended. For more information about data pumps, see *Administering Oracle GoldenGate for Windows and UNIX*.

1. In GGSCI on the source system, create the data-pump parameter file.

```
EDIT PARAMS name
```

Where *name* is the name of the data pump Extract.

2. Enter the data-pump parameters in the order shown, starting a new line for each parameter statement. Your input variables will be different. See Table 8-2 (page 8-4) for descriptions.

   Basic parameters for the data-pump Extract group:

```
EXTRACT extpump
USERIDALIAS tiger1
RMTHOST fin1, MGRPORT 7809 ENCRYPT AES192, KEYNAME securekey2
RMTTRAIL /ggs/dirdat/rt
SEQUENCE hr.employees_seq;
TABLE hr.*;
```

Table 8-2 Basic Parameters for a Data-pump Extract

Parameter	Description
`EXTRACT group`	*group* is the name of the data pump Extract. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.
`USERIDALIAS alias`	Specifies the alias of the database login credential of the user that is assigned to Extract. This credential must exist in the Oracle GoldenGate credential store. For more information, see Establishing Oracle GoldenGate Credentials (page 4-1).
`RMTHOST hostname, MGRPORT portnumber,[, ENCRYPT algorithm KEYNAME keyname]`	• `RMTHOST` specifies the name or IP address of the target system. • `MGRPORT` specifies the port number where Manager is running on the target. • `ENCRYPT` specifies optional encryption of data across TCP/IP. For additional options and encryption details, see *Reference for Oracle GoldenGate for Windows and UNIX*.
`RMTTRAIL pathname`	Specifies the path name of the remote trail. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.
`{TABLE | SEQUENCE} schema.object;`	Specifies a table or sequence, or multiple objects specified with a wildcard. In most cases, this listing will be the same as that in the primary Extract parameter file. • `TABLE` specifies a table or a wildcarded set of tables. • `SEQUENCE` specifies a sequence or a wildcarded set of sequences. • *schema* is the schema name or a wildcarded set of schemas. • *object* is the name of a table or sequence, or a wildcarded set of those objects. See *Administering Oracle GoldenGate for Windows and UNIX* for information about how to specify object names with and without wildcards. Terminate this parameter statement with a semi-colon. To exclude tables or sequences from a wildcard specification, use the `TABLEEXCLUDE` or `SEQUENCEEXCLUDE` parameter after the `TABLE` statement. For more information and for additional `TABLE` options that control data filtering, mapping, and manipulation, see *Reference for Oracle GoldenGate for Windows and UNIX*.

3. Enter any optional Extract parameters that are recommended for your configuration. You can edit this file at any point before starting processing by using the `EDIT PARAMS` command in GGSCI. For more information, see the *Reference for Oracle GoldenGate for Windows and UNIX*.
4. See Optional Parameters for Integrated Modes (page A-1) for additional configuration considerations.

5. Save and close the file.

8.5 Next Steps

Once you have created a basic parameter file for classic capture, see the following for related configuration steps.

Configuring Oracle GoldenGate Apply (page 9-1)

Additional Oracle GoldenGate Configuration Considerations (page 10-1)

Additional Configuration Steps for Using Classic Capture (page 11-1)

Installing Trigger-Based DDL Capture (page D-1) (to use Oracle GoldenGate DDL support)

Configuring DDL Support (page 13-1) (to use Oracle GoldenGate DDL support)

Creating Process Groups (page 14-1)

Instantiating Oracle GoldenGate Replication (page 15-1)

Supporting Changes to XML Schemas (page E-1)

9
Configuring Oracle GoldenGate Apply

This chapter contains instructions for configuring the Replicat apply process in either nonintegrated or integrated mode. See Choosing Capture and Apply Modes (page 5-1) for more information about Extract modes.

Topics:

9.1 Prerequisites for Configuring Replicat

The guidelines in the following sections should be satisfied before configuring Extract in integrated mode.

1. Preparing the Database for Oracle GoldenGate (page 3-1)
2. Establishing Oracle GoldenGate Credentials (page 4-1)
3. Choosing Capture and Apply Modes (page 5-1)
4. Create the Oracle GoldenGate instance on the target system by configuring the Manager process. See *Administering Oracle GoldenGate for Windows and UNIX*.
5. Additionally, review the guidelines in *Administering Oracle GoldenGate for Windows and UNIX*.

> **Note:**
>
> To switch an active Replicat configuration from one mode to the other, perform these configuration steps and then see *Administering Oracle GoldenGate for Windows and UNIX*.

9.2 What to Expect from these Instructions

These instructions show you how to configure a basic Replicat parameter (configuration) file. Your business requirements probably will require a more complex topology, but this procedure forms a basis for the rest of your configuration steps.

By performing these steps, you can:

- get the basic configuration file established.

- build upon it later by adding more parameters as you make decisions about features or requirements that apply to your environment.
- use copies of it to make the creation of additional Replicat parameter files faster than starting from scratch.

> **Note:**
>
> These instructions do not configure Replicat to apply DDL to the target. To support DDL, create the basic Replicat parameter file and then see Configuring DDL Support (page 13-1) for configuration instructions.

9.3 Creating a Checkpoint Table (Nonintegrated Replicat Only)

The checkpoint table is a required component of nonintegrated Replicat. It is not required for integrated Replicat and is ignored during runtime if one is used.

A nonintegrated Replicat maintains its recovery checkpoints in the checkpoint table, which is stored in the target database. Checkpoints are written to the checkpoint table within the Replicat transaction. Because a checkpoint either succeeds or fails with the transaction, Replicat ensures that a transaction is only applied once, even if there is a failure of the process or the database.

> **Note:**
>
> This procedure installs a default checkpoint table, which is sufficient in most cases. More than one checkpoint table can be used, such as to use a different one for each Replicat group. To use a non-default checkpoint table, which overrides the default table, use the `CHECKPOINTTABLE` option of `ADD REPLICAT` when you create Replicat processes in the steps in Instantiating Oracle GoldenGate Replication (page 15-1). For details, see *Reference for Oracle GoldenGate for Windows and UNIX*.

9.3.1 Adding the Checkpoint Table to the Target Database

1. From the Oracle GoldenGate directory on the target, run GGSCI and issue the `DBLOGIN` command to log into the target database.

   ```
   DBLOGIN USERIDALIAS alias
   ```

   Where:

- *alias* specifies the alias of the database login credential of a user that can create tables in a schema that is accessible to Replicat. This credential must exist in the Oracle GoldenGate credential store. For more information, see Establishing Oracle GoldenGate Credentials (page 4-1).

2. In GGSCI, create the checkpoint table in a schema of your choice (ideally dedicated to Oracle GoldenGate).

   ```
   ADD CHECKPOINTTABLE [container.]schema.table
   ```

   Where:

   - *container* is the name of the container if *schema.table* is in a multitenant container database. This container can be the root container or a pluggable database that contains the table.
   - *schema.table* are the schema and name of the table. See *Administering Oracle GoldenGate for Windows and UNIX* for instructions for specifying object names.

9.3.2 Specifying the Checkpoint Table in the Oracle GoldenGate Configuration

To specify the checkpoint table in the Oracle GoldenGate configuration:

1. Create a GLOBALS file (or edit the existing one).

   ```
   EDIT PARAMS ./GLOBALS
   ```

   > **Note:**
   >
   > EDIT PARAMS creates a simple text file. When you save the file after EDIT PARAMS, it is saved with the name GLOBALS in upper case, without a file extension. It must remain as such, and the file must remain in the root Oracle GoldenGate directory.

2. In the GLOBALS file, enter the CHECKPOINTTABLE parameter.

   ```
   CHECKPOINTTABLE [container.]schema.table
   ```

3. Save and close the GLOBALS file.

9.3.3 Disabling Default Asynchronous COMMIT to Checkpoint Table

When a nonintegrated Replicat uses a checkpoint table, it uses an asynchronous COMMIT with the NOWAIT option to improve performance. Replicat can continue processing immediately after applying this COMMIT, while the database logs the transaction in the background. You can disable the asynchronous COMMIT with NOWAIT by using the DBOPTIONS parameter with the DISABLECOMMITNOWAIT option in the Replicat parameter file.

> **Note:**
>
> When the configuration of a nonintegrated Replicat group does not include a checkpoint table, the checkpoints are maintained in a file on disk. In this case, Replicat uses `COMMIT` with `WAIT` to prevent inconsistencies in the event of a database failure that causes the state of the transaction, as in the checkpoint file, to be different than its state after the recovery.

9.4 Configuring Replicat

These steps configure the Replicat process. Integrated and nonintegrated Replicat modes are taken into consideration. For more advanced mapping options, see *Administering Oracle GoldenGate for Windows and UNIX*.

> **Note:**
>
> To configure Replicat against a pluggable database, configure a Replicat process in integrated apply mode

1. In GGSCI on the target system, create the Replicat parameter file.

   ```
   EDIT PARAMS name
   ```

   Where: *name* is the name of the Replicat group.

2. Enter the Replicat parameters in the order shown, starting a new line for each parameter statement. See Table 9-1 (page 9-4) for descriptions.

   Basic parameters for the Replicat group in nonintegrated mode:

   ```
   REPLICAT financer
   USERIDALIAS tiger2
   ASSUMETARGETDEFS
   MAP hr.*, TARGET hr2.*;
   ```

   Basic parameters for the Replicat group in integrated Replicat mode:

   ```
   REPLICAT financer
   DBOPTIONS INTEGRATEDPARAMS(parallelism 6)
   USERIDALIAS tiger2
   ASSUMETARGETDEFS
   MAP hr.*, TARGET hr2.*;
   ```

Table 9-1 Basic Parameters for Replicat

Parameter	Description
`REPLICAT group`	*group* is the name of the Replicat group.

Table 9-1 (Cont.) Basic Parameters for Replicat

Parameter	Description
`DBOPTIONS DEFERREFCONST`	Applies to Replicat in nonintegrated mode. `DEFERREFCONST` sets constraints to `DEFERRABLE` to delay the enforcement of cascade constraints by the target database until the Replicat transaction is committed. See *Reference for Oracle GoldenGate for Windows and UNIX* for additional important information.
`DBOPTIONS INTEGRATEDPARAMS (parameter[, ...])`	This parameter specification applies to Replicat in integrated mode. It specifies optional parameters for the inbound server. See Optional Parameters for Integrated Modes (page A-1)for additional important information about these `DBOPTIONS` options.
`USERIDALIAS alias`	Specifies the alias of the database login credential of the user that is assigned to Replicat. This credential must exist in the Oracle GoldenGate credential store. For more information, see Establishing Oracle GoldenGate Credentials (page 4-1)
`SOURCECATALOG container`	Use this parameter when the source database is a multitenant container database. Specifies the name of a pluggable database that is to be used as the default container for all subsequent `MAP` parameters that contain two-part names. This parameter enables you to use two-part object names (`schema.object`) rather than three-part names (`container.schema.object`). It remains in effect until another `SOURCECATALOG` parameter is encountered or a full three-part `MAP` specification is encountered. See *Reference for Oracle GoldenGate for Windows and UNIX* for more information about `SOURCECATALOG`.
`MAP [container.]schema.object, TARGET schema.object;`	Specifies the relationship between a source table or sequence, or multiple objects, and the corresponding target object or objects. • `MAP` specifies the source table or sequence, or a wildcarded set of objects. • `TARGET` specifies the target table or sequence or a wildcarded set of objects. • *`container`* is the name of a container, if the source database is a multitenant container database. • *`schema`* is the schema name or a wildcarded set of schemas. • *`object`* is the name of a table or sequence, or a wildcarded set of objects. Terminate this parameter statement with a semi-colon. To exclude objects from a wildcard specification, use the `MAPEXCLUDE` parameter. For more information and for additional options that control data filtering, mapping, and manipulation, see `MAP` in *Reference for Oracle GoldenGate for Windows and UNIX*.

3. If using integrated Replicat, add the following parameters to the Extract parameter file:

   - `LOGALLSUPCOLS`: This parameter ensures the capture of the supplementally logged columns in the before image. This parameter is valid for any source database that is supported by Oracle GoldenGate. For Extract versions older than 12c, you can use `GETUPDATEBEFORES` and `NOCOMPRESSDELETES` parameters to satisfy the same requirement. The database must be configured to log the before and after values of the primary key, unique indexes, and foreign keys. See *Reference for Oracle GoldenGate for Windows and UNIX* for more information about `LOGALLSUPCOLS`.

- The `UPDATERECORDFORMAT` parameter set to COMPACT: This setting causes Extract to combine the before and after images of an `UPDATE` operation into a single record in the trail. This parameter is valid for Oracle Database versions 11.2.0.4 and later and improves the performance of an integrated Replicat. See *Reference for Oracle GoldenGate for Windows and UNIX* for more information.

4. Enter any optional Replicat parameters that are recommended for your configuration. You can edit this file at any point before starting processing by using the `EDIT PARAMS` command in GGSCI. For more information, see the *Reference for Oracle GoldenGate for Windows and UNIX*.
5. See Optional Parameters for Integrated Modes (page A-1) for additional configuration considerations.
6. Save and close the file.

> **Note:**
>
> See *Administering Oracle GoldenGate for Windows and UNIX* for important information about making configuration changes to Replicat once processing is started, if using integrated Replicat.

9.5 Next Steps

Once you have created a basic parameter file for Replicat, see the following for additional configuration steps.

Configuring Capture in Classic Mode (page 8-1) or Configuring Capture in Integrated Mode (page 7-1) if you have not configured capture yet.

Additional Configuration Steps For Using Nonintegrated Replicat (page 12-1) (if using nonintegrated Replicat)

Additional Oracle GoldenGate Configuration Considerations (page 10-1)

Configuring DDL Support (page 13-1) (to use Oracle GoldenGate DDL support)

Creating Process Groups (page 14-1)

Instantiating Oracle GoldenGate Replication (page 15-1)

10 Additional Oracle GoldenGate Configuration Considerations

This chapter contains additional configuration considerations that may apply to your database environment.

Topics:

10.1 Ensuring Row Uniqueness in Source and Target Tables

Oracle GoldenGate requires a unique row identifier on the source and target tables to locate the correct target rows for replicated updates and deletes. Unless a `KEYCOLS` clause is used in the `TABLE` or `MAP` statement, Oracle GoldenGate selects a row identifier to use in the following order of priority, depending on the number and type of constraints that were logged (see Configuring Logging Properties (page 3-2)).

1. Primary key if it does not contain any extended (32K) `VARCHAR2/NVARCHAR2` columns. Primary key without invisible columns.
2. Unique key. Unique key without invisible columns.

   In the case of a nonintegrated Replicat, the selection of the unique key is as follows:

   - First unique key alphanumerically with no virtual columns, no UDTs, no function-based columns, no nullable columns, and no extended (32K) `VARCHAR2/NVARCHAR2` columns. To support a key that contains columns that are part of an invisible index, you must use the `ALLOWINVISIBLEINDEXKEYS` parameter in the Oracle GoldenGate `GLOBALS` file.
   - First unique key alphanumerically with no virtual columns, no UDTs, no extended (32K) `VARCHAR2/NVARCHAR2` columns, or no function-based columns, but can include nullable columns. To support a key that contains columns that are part of an invisible index, you must use the `ALLOWINVISIBLEINDEXKEYS` parameter in the Oracle GoldenGate `GLOBALS` file.
3. If none of the preceding key types exist (even though there might be other types of keys defined on the table) Oracle GoldenGate constructs a pseudo key of all columns that the database allows to be used in a unique key, excluding virtual

columns, UDTs, function-based columns, extended (32K) VARCHAR2/NVARCHAR2 columns, and any columns that are explicitly excluded from the Oracle GoldenGate configuration by an Oracle GoldenGate user.

Unless otherwise excluded due to the preceding restrictions, invisible columns are allowed in the pseudo key.

> **Note:**
>
> If there are other, non-usable keys on a table or if there are no keys at all on the table, Oracle GoldenGate logs an appropriate message to the report file. Constructing a key from all of the columns impedes the performance of Oracle GoldenGate on the source system. On the target, this key causes Replicat to use a larger, less efficient WHERE clause.

If a table does not have an appropriate key, or if you prefer the existing key(s) not to be used, you can define a substitute key if the table has columns that always contain unique values. You define this substitute key by including a KEYCOLS clause within the Extract TABLE parameter and the Replicat MAP parameter. The specified key will override any existing primary or unique key that Oracle GoldenGate finds. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.

10.2 Installing Support for Oracle Sequences

To support Oracle sequences, you must install some database procedures. These procedures support the Oracle GoldenGate FLUSH SEQUENCE command, which you issue immediately after you start the Oracle GoldenGate processes for the first time (typically when you perform the initial data synchronization procedure).

To Install Oracle Sequence Objects

You will perform steps on the source and target systems.

1. In SQL*Plus, connect to the source and target Oracle systems as SYSDBA.
2. If you already assigned a database user to support the Oracle GoldenGate DDL replication feature, you can skip this step. Otherwise, in SQL*Plus on both systems create a database user that can also be the DDL user.

   ```
   CREATE USER DDLuser IDENTIFIED BY password;
   GRANT CONNECT, RESOURCE, DBA TO DDLuser;
   ```

3. From the Oracle GoldenGate installation directory on each system, run GGSCI.
4. In GGSCI, issue the following command on each system.

   ```
   EDIT PARAMS ./GLOBALS
   ```

5. In each GLOBALS file, enter the GGSCHEMA parameter and specify the schema of the DDL user that you created earlier in this procedure.

   ```
   GGSCHEMA schema
   ```

6. Save and close the files.
7. In SQL*Plus on both systems, run the sequence.sql script from the root of the Oracle GoldenGate installation directory. This script creates some procedures for

use by Oracle GoldenGate processes. (Do not run them yourself.) You are prompted for the user information that you created in the first step.

```
@sequence.sql
```

8. In SQL*Plus on the source system, grant `EXECUTE` privilege on the `updateSequence` procedure to a database user that can be used to issue the `DBLOGIN` command. Remember or record this user. You use `DBLOGIN` to log into the database prior to issuing the `FLUSH SEQUENCE` command, which calls the procedure.

```
GRANT EXECUTE on DDLuser.updateSequence TO DBLOGINuser;
```

9. In SQL*Plus on the target system, grant `EXECUTE` privilege on the `replicateSequence` procedure to the Replicat database user.

```
GRANT EXECUTE on DDLuser.replicateSequence TO Replicatuser;
```

10. In SQL*Plus on the source system, issue the following statement in SQL*Plus.

```
ALTER TABLE sys.seq$ ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS;
```

10.3 Handling Special Data Types

This section applies whether Extract operates in classic or integrated capture mode, unless otherwise noted. It addresses special configuration requirements for the following Oracle data types:

- Multibyte Character Types (page 10-3)
- Oracle Spatial Objects (page 10-4)
- TIMESTAMP (page 10-4)
- Large Objects (LOB) (page 10-5)
- XML (page 10-5)
- User Defined Types (page 10-6)

10.3.1 Multibyte Character Types

Multi-byte characters are supported as part of a supported character set. If the semantics setting of an Oracle source database is `BYTE` and the setting of an Oracle target is `CHAR`, use the Replicat parameter `SOURCEDEFS` in your configuration, and place a definitions file that is generated by the `DEFGEN` utility on the target. These steps are required to support the difference in semantics, whether or not the source and target data definitions are identical. Replicat refers to the definitions file to determine the upper size limit for fixed-size character columns.

For more information about character-set support, see *Administering Oracle GoldenGate for Windows and UNIX*..

For information about SOURCEDEFS and the DEFGEN utility, see *Administering Oracle GoldenGate for Windows and UNIX*.

10.3.2 Oracle Spatial Objects

To replicate tables that contain one or more columns of SDO_GEORASTER object type from an Oracle source to an Oracle target, follow these instructions to configure Oracle GoldenGate to process them correctly.

1. Create a TABLE statement and a MAP statement for the georaster tables and also for the related raster data tables.
2. If the METADATA attribute of the SDO_GEORASTER data type in any of the values exceeds 1 MB, use the DBOPTIONS parameter with the XMLBUFSIZE option to increase the size of the memory buffer that stores the embedded SYS.XMLTYPE attribute of the SDO_GEORASTER data type. If the buffer is too small, Extract abends. See XMLBUFSIZE in *Reference for Oracle GoldenGate for Windows and UNIX*.
3. To ensure the integrity of the target georaster tables and the spatial data, keep the trigger enabled on both source and target. Use the REPERROR option of the MAP parameter to handle the "ORA-01403 No data found" error that occurs as a result of keeping the trigger enabled on the target. It occurs when a row in the source georaster table is deleted, and the trigger cascades the delete to the raster data table. Both deletes are replicated. The replicated parent delete triggers the cascaded (child) delete on the target. When the replicated child delete arrives, it is redundant and generates the error. To use REPERROR, do the following:
   - Use a REPERROR statement in each MAP statement that contains a raster data table.
   - Use Oracle error 1403 as the SQL error.
   - Use any of the response options as the error handling.

A sufficient way to handle the errors on raster tables caused by active triggers on target georaster tables is to use REPERROR with DISCARD to discard the cascaded delete that triggers them. The trigger on the target georaster table performs the delete to the raster data table, so the replicated one is not needed.

```
MAP geo.st_rdt, TARGET geo.st_rdt, REPERROR (-1403, DISCARD) ;
```

If you need to keep an audit trail of the error handling, use REPERROR with EXCEPTION to invoke exceptions handling. For this, you create an exceptions table and map the source raster data table twice:

- once to the actual target raster data table (with REPERROR handling the 1403 errors).
- again to the exceptions table, which captures the 1403 error and other relevant information by means of a COLMAP clause.

For more information about using an exceptions table, see *Administering Oracle GoldenGate for Windows and UNIX*.

For more information about REPERROR options, see *Reference for Oracle GoldenGate for Windows and UNIX*.

10.3.3 TIMESTAMP

To replicate timestamp data, follow these guidelines.

1. To prevent Oracle GoldenGate from abending on `TIMESTAMP WITH TIME ZONE` as `TZR`, use the Extract parameter `TRANLOGOPTIONS` with one of the following:
   - `INCLUDEREGIONID` to replicate `TIMESTAMP WITH TIME ZONE` as `TZR` from an Oracle source to an Oracle target of the same version or later.
   - `INCLUDEREGIONIDWITHOFFSET` to replicate `TIMESTAMP WITH TIMEZONE` as `TZR` from an Oracle source that is at least v10g to an earlier Oracle target, or from an Oracle source to a non-Oracle target.

   These options allow replication to Oracle versions that do not support `TIMESTAMP WITH TIME ZONE` as `TZR` and to database systems that only support time zone as a UTC offset. .
2. Oracle Database normalizes `TIMESTAMP WITH LOCAL TIME ZONE` data to the local time zone of the database that receives it, the target database in the case of Oracle GoldenGate. To preserve the original time stamp of the data that it applies, Replicat sets its session to the time zone of the source database. You can override this default and supply a different time zone by using the `SOURCETIMEZONE` parameter in the Replicat parameter file. To force Replicat to set its session to the target time zone, use the `PRESERVETARGETTIMEZONE` parameter.

   You can also use the `SOURCETIMEZONE` parameter to specify the source time zone for data that is captured by an Extract that is earlier than version 12.1.2. Those versions do not write the source time zone to the trail.

10.3.4 Large Objects (LOB)

The following are some configuration guidelines for LOBs in both classic capture and integrated capture mode.

1. Store large objects out of row if possible.
2. (Applies only to integrated capture) Integrated capture captures LOBs from the redo log. For `UPDATE` operations on a LOB document, only the changed portion of the LOB is logged. To force whole LOB documents to be written to the trail when only the changed portion is logged, use the `TRANLOGOPTIONS` parameter with the `FETCHPARTIALLOB` option in the Extract parameter file. When Extract receives partial LOB content from the logmining server, it fetches the full LOB image instead of processing the partial LOB. Use this option when replicating to a non-Oracle target or in other conditions where the full LOB image is required.

10.3.5 XML

The following are tools for working with XML within Oracle GoldenGate constraints.

- Although both classic and integrated capture modes do not support the capture of changes made to an XML schema, you may be able to evolve the schemas and then resume replication of them without the need for a resynchronization. See Supporting Changes to XML Schemas (page E-1).
- (Applies only to integrated capture) Integrated capture captures XML from the redo log. For `UPDATE` operations on an XML document, only the changed portion of the XML is logged if it is stored as `OBJECT RELATIONAL` or `BINARY`. To force whole XML documents to be written to the trail when only the changed portion is logged, use the `TRANLOGOPTIONS` parameter with the `FETCHPARTIALXML` option in the Extract parameter file. When Extract receives partial XML content from the logmining server, it fetches the full XML document instead of processing the partial XML.

Use this option when replicating to a non-Oracle target or in other conditions where the full XML image is required.

10.3.6 User Defined Types

If Oracle Database is compatible with releases greater than or equal to 12.0.0.0.0, then integrated Extract captures data from redo (no fetch). For more information, see Setting Flashback Query (page 3-8).

If replicating source data that contains user-defined types with the `NCHAR`, `NVARCHAR2`, or `NCLOB` attribute to an Oracle target, use the `HAVEUDTWITHNCHAR` parameter in the Replicat parameter file. When this type of data is encountered in the trail, `HAVEUDTWITHNCHAR` causes Replicat to connect to the Oracle target in `AL32UTF8`, which is required when a user-defined data type contains one of those attributes. `HAVEUDTWITHNCHAR` is required even if `NLS_LANG` is set to `AL32UTF8` on the target. By default Replicat ignores `NLS_LANG` and connects to an Oracle Database in the native character set of the database. Replicat uses the `OCIString` object of the Oracle Call Interface, which does not support `NCHAR`, `NVARCHAR2`, or `NCLOB` attributes, so Replicat must bind them as `CHAR`. Connecting to the target in `AL32UTF8` prevents data loss in this situation. `HAVEUDTWITHNCHAR` must appear before the `USERID` or `USERIDALIAS` parameter in the parameter file.

10.4 Handling Other Database Properties

The following table shows database properties that may affect Oracle GoldenGate and the parameters that you can use to resolve or work around the condition.

Table 10-1 Handling Other Database Properties

Database Property	Concern/Resolution
Table with interval partitioning	To support tables with interval partitioning, make certain that the `WILDCARDRESOLVE` parameter remains at its default of `DYNAMIC`.
Table with virtual columns	Virtual columns are not logged, and Oracle does not permit DML on virtual columns. You can, however, capture this data and map it to a target column that is not a virtual column by doing the following: Include the table in the Extract `TABLE` statement and use the `FETCHCOLS` option of `TABLE` to fetch the value from the virtual column in the database. In the Replicat `MAP` statement, map the source virtual column to the non-virtual target column.
Table with inherently updateable view	To replicate to an inherently updateable view, define a key on the unique columns in the updateable view by using a `KEYCOLS` clause in the same `MAP` statement in which the associated source and target tables are mapped.
Redo logs or archives in different locations	The `TRANLOGOPTIONS` parameter contains options to handle environments where the redo logs or archives are stored in a different location than the database default or on a different platform from that on which Extract is running. These options may be required when Extract operates in classic capture mode. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.

Table 10-1 (Cont.) Handling Other Database Properties

Database Property	Concern/Resolution
TRUNCATE operations	To replicate TRUNCATE operations, choose one of two options: • Standalone TRUNCATE support by means of the GETTRUNCATES parameter replicates TRUNCATE TABLE, but no other TRUNCATE options. Use only if not using Oracle GoldenGate DDL support. • The full DDL support replicates TRUNCATE TABLE, ALTER TABLE TRUNCATE PARTITION, and other DDL. To install this support, see Installing Trigger-Based DDL Capture (page D-1)..
Sequences	To replicate DDL for sequences (CREATE, ALTER, DROP, RENAME), use Oracle GoldenGate DDL support. To replicate just sequence values, use the SEQUENCE parameter in the Extract parameter file. This does *not* require the Oracle GoldenGate DDL support environment. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.

10.5 Controlling the Checkpoint Frequency

The CHECKPOINTRETENTIONTIME option of the TRANLOGOPTIONS parameter controls the number of days that Extract in integrated mode retains checkpoints before purging them automatically. Partial days can be specified using decimal values. For example, 8.25 specifies 8 days and 6 hours. The default is seven days. For more information about this parameter, see *Reference for Oracle GoldenGate for Windows and UNIX*.

10.6 Excluding Replicat Transactions

In a bidirectional configuration, Replicat must be configured to mark its transactions, and Extract must be configured to exclude Replicat transactions so that they do not propagate back to their source. There are two methods to accomplish this as follows:

Method 1

Valid only for Oracle to Oracle implementations.

When Extract is in classic or integrated mode (Replicat can be in either integrated or nonintegrated mode), use the following parameters:

- Use DBOPTIONS with the SETTAG option in the Replicat parameter file. The inbound server tags the transactions of that Replicat with the specified value, which identifies those transactions in the redo stream. The default value for SETTAG is 00.
- Use the TRANLOGOPTIONS parameter with the EXCLUDETAG option in a classic or integrated Extract parameter file. The logmining server associated with that Extract excludes redo that is tagged with the SETTAG value. Multiple EXCLUDETAG statements can be used to exclude different tag values, if desired.

  For Oracle to Oracle, this is the recommended method.

Method 2

Valid for any implementation; Oracle or heterogeneous database configurations.

Alternatively, when Extract is in classic or integrated capture mode, you could also use the Extract `TRANLOGOPTIONS` parameter with the `EXCLUDEUSER` or `EXCLUDEUSERID` option to ignore Replicat the DDL and DML transactions based on its user name or ID. Multiple `EXCLUDEUSER` statements can be used. The specified user is subject to the rules of the `GETREPLICATES` or `IGNOREREPLICATES` parameter.

For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.

10.7 Advanced Configuration Options for Oracle GoldenGate

You may need to configure Oracle GoldenGate with advanced options to suit your business needs. See the following:

- For additional configuration guidelines to achieve specific replication topologies, see *Administering Oracle GoldenGate for Windows and UNIX*. This guide includes instructions for the following configurations:
  - Using Oracle GoldenGate for live reporting
  - Using Oracle GoldenGate for real-time data distribution
  - Configuring Oracle GoldenGate for real-time data warehousing
  - Using Oracle GoldenGate to maintain a live standby database
  - Using Oracle GoldenGate for active-active high availability

  That guide also contains information about:
  - Oracle GoldenGate architecture
  - Oracle GoldenGate commands
  - Oracle GoldenGate initial load methods
  - Configuring security
  - Using customization features
  - Configuring data filtering and manipulation
- If either the source or target database is non-Oracle, follow the installation and configuration instructions in the Oracle GoldenGate installation and setup guide for that database, and then refer to the Oracle GoldenGate administration and reference documentation for further information.

11

Additional Configuration Steps for Using Classic Capture

This chapter contains additional configuration and preparation requirements that are specific only to Extract when operating in *classic capture* mode.
These requirements supplement the basic configuration requirements documented in Configuring Capture in Classic Mode (page 8-1)

Topics:

11.1 Configuring Oracle TDE Data in Classic Capture Mode

The following special configuration steps are required to support TDE when Extract is in classic capture mode. This section *does not* apply to Extract in integrated capture mode.

> **Note:**
>
> When in integrated mode, Extract leverages the database logmining server and supports TDE column encryption and TDE tablespace encryption without special setup requirements or parameter settings. For more information about integrated capture, see Choosing Capture and Apply Modes (page 5-1).

- Updating the Oracle Shared Secret in the Parameter File (page 11-6)

11.1.1 Overview of TDE Support in Classic Capture Mode

TDE support when Extract is in classic capture mode requires the exchange of two kinds of keys:

- The *encrypted key* can be a table key (column-level encryption), an encrypted redo log key (tablespace-level encryption), or both. This key is shared between the Oracle Database and Extract.
- The *decryption key* is named `ORACLEGG` and its password is known as the *shared secret*. This key is stored securely in the Oracle and Oracle GoldenGate domains. Only a party that has possession of the shared secret can decrypt the table and redo log keys.

The encrypted keys are delivered to the Extract process by means of built-in PL/SQL code. Extract uses the shared secret to decrypt the data. Extract never handles the wallet master key itself, nor is it aware of the master key password. Those remain within the Oracle Database security framework.

Extract never writes the decrypted data to any file other than a trail file, not even a discard file (specified with the `DISCARDFILE` parameter). The word "`ENCRYPTED`" will be written to any discard file that is in use.

The impact of this feature on Oracle GoldenGate performance should mirror that of the impact of decryption on database performance. Other than a slight increase in Extract startup time, there should be a minimal affect on performance from replicating TDE data.

11.1.2 Requirements for Capturing TDE in Classic Capture Mode

The following are requirements for Extract to support TDE capture:

- To maintain high security standards, the Oracle GoldenGate Extract process should run as part of the `oracle` user (the user that runs the Oracle Database). That way, the keys are protected in memory by the same privileges as the `oracle` user.
- The Extract process must run on the same machine as the database installation.
- Even if using TDE with a Hardware Security Module, you must use a software wallet. Instructions are provided in Oracle Security Officer Tasks (page 11-3) in the configuration steps for moving from an HSM-only to an HSM-plus-wallet configuration and configuring the `sqlnet.ora` file correctly.
- Whenever the source database is upgraded, you must rekey the master key.

11.1.3 Required Database Patches for TDE Support

To support TDE on Oracle 11.2.0.2, refer to article 1557031.1 on the My Oracle Support website (`https://support.oracle.com`).

11.1.4 Configuring Classic Capture for TDE Support

The following outlines the steps that the Oracle Security Officer and the Oracle GoldenGate Administrator take to establish communication between the Oracle server and the Extract process.

11.1.4.1 Agree on a Shared Secret that Meets Oracle Standards

Agree on a *shared secret* password that meets or exceeds Oracle password standards. This password must not be known by anyone else. For guidelines on creating secure passwords, see *Oracle Database Security Guide*.

11.1.4.2 Oracle DBA Tasks

1. Log in to SQL*Plus as a user with the `SYSDBA` system privilege. For example:

   ```
   sqlplus sys/as sysdba
   Connected.
   Enter password: password
   ```

2. Run the `prvtclkm.plb` file that is installed in the Oracle `admin` directory. The `prvtclkm.plb` file creates the `DBMS_INTERNAL_CLKM` PL/SQL package, which enables Oracle GoldenGate to extract encrypted data from an Oracle Database.

   ```
   @?/app/oracle/product/orcl111/rdbms/admin/prvtclkm.plb
   ```

3. Grant `EXEC` privilege on `DBMS_INTERNAL_CLKM` PL/SQL package to the Extract database user.

   ```
   GRANT EXECUTE ON DBMS_INTERNAL_CLKM TO psmith;
   ```

4. Exit SQL*Plus.

11.1.4.3 Oracle Security Officer Tasks

1. Oracle GoldenGate requires the use of a software wallet even with HSM. If you are currently using HSM-only mode, move to HSM-plus-wallet mode by taking the following steps:

   a. Change the `sqlnet.ora` file configuration as shown in the following example, where the wallet directory can be any location on disk that is accessible (rwx) by the owner of the Oracle Database. This example shows a best-practice location, where `my_db` is the `$ORACLE_SID`.

      ```
      ENCRYPTION_WALLET_LOCATION=
       (SOURCE=(METHOD=HSM)(METHOD_DATA=
         (DIRECTORY=/etc/oracle/wallets/my_db)))
      ```

   b. Log in to `orapki` (or Wallet Manager) as the owner of the Oracle Database, and create an auto-login wallet in the location that you specified in the `sqlnet.ora` file. When prompted for the wallet password, specify the **same**

password as the HSM password (or HSM Connect String). These two passwords must be identical.

```
cd /etc/oracle/wallets/my_db
orapki wallet create -wallet . -auto_login[_local]
```

> **Note:**
>
> The Oracle Database owner must have full operating system privileges on the wallet.

c. Add the following entry to the empty wallet to enable an 'auto-open' HSM:

```
mkstore -wrl . -createEntry ORACLE.TDE.HSM.AUTOLOGIN non-empty-string
```

2. Create an entry named `ORACLEGG` in the wallet. **`ORACLEGG` must be the name of this key.** The password for this key must be the agreed-upon shared secret, but do not enter this password on the command line. Instead, wait to be prompted.

```
mkstore -wrl ./ -createEntry ORACLE.SECURITY.CL.ENCRYPTION.ORACLEGG
Oracle Secret Store Tool : Version 11.2.0.3.0 - Production
Copyright (c) 2004, 2011, Oracle and/or its affiliates. All rights reserved.
Your secret/Password is missing in the command line
Enter your secret/Password: sharedsecret
Re-enter your secret/Password: sharedsecret
Enter wallet password: hsm/wallet_password
```

3. Verify the `ORACLEGG` entry.

```
mkstore -wrl . -list
Oracle Secret Store Tool : Version 11.2.0.3.0 - Production
Copyright (c) 2004, 2011, Oracle and/or its affiliates. All rights reserved.
Enter wallet password: hsm/wallet_password
Oracle Secret Store entries:
ORACLE.SECURITY.CL.ENCRYPTION.ORACLEGG
```

4. Log in to SQL*Plus as a user with the `SYSDBA` system privilege.

5. Close and then re-open the wallet.

```
SQL> alter system set encryption wallet close identified by "hsm/
wallet_password";
System altered.
SQL> alter system set encryption wallet open identified by "hsm/
wallet_password";
System altered.
```

This inserts the password into the auto-open wallet, so that no password is required to access encrypted data with the TDE master encryption key stored in HSM.

6. Switch log files.

```
alter system switch logfile;
System altered.
```

7. If this is an Oracle RAC environment and you are using copies of the wallet on each node, make the copies now and then reopen each wallet.

> **Note:**
>
> Oracle recommends using one wallet in a shared location, with synchronized access among all Oracle RAC nodes.

11.1.4.4 Oracle GoldenGate Administrator Tasks

1. Run GGSCI.
2. Issue the `ENCRYPT PASSWORD` command to encrypt the shared secret so that it is obfuscated within the Extract parameter file. *This is a security requirement*.

   ```
   ENCRYPT PASSWORD sharedsecret {AES128 | AES192 | AES256} ENCRYPTKEY keyname
   ```

   Where:

   - *sharedsecret* is the clear-text shared secret. This value is case-sensitive.
   - `{AES128 | AES192 | AES256}` specifies Advanced Encryption Standard (AES) encryption. Specify one of the values, which represents the desired key length.
   - *keyname* is the logical name of the encryption key in the `ENCKEYS` lookup file. Oracle GoldenGate uses this key to look up the actual key in the `ENCKEYS` file. To create a key and `ENCKEYS` file, see *Administering Oracle GoldenGate for Windows and UNIX*.

     Example:

     ```
     ENCRYPT PASSWORD sharedsecret AES256 ENCRYPTKEY mykey1
     ```
3. In the Extract parameter file, use the `DBOPTIONS` parameter with the `DECRYPTPASSWORD` option. As input, supply the encrypted shared secret and the decryption key.

   ```
   DBOPTIONS DECRYPTPASSWORD sharedsecret {AES128 | AES192 | AES256} ENCRYPTKEY
   keyname
   ```

   Where:

   - *sharedsecret* is the encrypted shared secret.
   - `{AES128 | AES192 | AES256}` must be same value that was used for `ENCRYPT PASSWORD`.
   - *keyname* is the logical name of the encryption key in the `ENCKEYS` lookup file.

     Example:

     ```
     DBOPTIONS DECRYPTPASSWORD AACAAAAAAAAAAAIALCKDZIRHOJBHOJUH AES256 ENCRYPTKEY
     mykey1
     ```
4. Log in to SQL*Plus as a user with the `SYSDBA` system privilege.
5. Close and then re-open the wallet.

   ```
   SQL> alter system set encryption wallet close identified by "hsm/
   wallet_password";
   System altered.
   SQL> alter system set encryption wallet open identified by "hsm/
   wallet_password";
   System altered.
   ```

11.1.5 Recommendations for Maintaining Data Security after Decryption

Extract decrypts the TDE data and writes it to the trail as clear text. To maintain data security throughout the path to the target database, it is recommended that you also deploy Oracle GoldenGate security features to:

- encrypt the data in the trails
- encrypt the data in transit across TCP/IP

For more information, see *Administering Oracle GoldenGate for Windows and UNIX*.

11.1.6 Performing DDL while TDE Capture is Active

If DDL will ever be performed on a table that has column-level encryption, or if table keys will ever be re-keyed, you must either quiesce the table while the DDL is performed or enable Oracle GoldenGate DDL support. It is more practical to have the DDL environment active so that it is ready, because a re-key usually is a response to a security violation and must be performed immediately. To install the Oracle GoldenGate DDL environment, see Installing Trigger-Based DDL Capture (page D-1). To configure Oracle GoldenGate DDL support, see Configuring DDL Support (page 13-1). For tablespace-level encryption, the Oracle GoldenGate DDL support is not required.

11.1.7 Rekeying after a Database Upgrade

Whenever the source database is upgraded and Oracle GoldenGate is capturing TDE data, you must rekey the master key, and then restart the database and Extract. The commands to rekey the master key are:

```
alter system set encryption key identified by "mykey";
```

11.1.8 Updating the Oracle Shared Secret in the Parameter File

Use this procedure to update and encrypt the TDE shared secret within the Extract parameter file.

1. Run GGSCI.
2. Stop the Extract process.

   ```
   STOP EXTRACT group
   ```

3. Modify the `ORACLEGG` entry in the Oracle wallet. `ORACLEGG` must remain the name of the key. For instructions, see *Oracle Database Advanced Security Guide*.
4. Issue the `ENCRYPT PASSWORD` command to encrypt the new shared secret.

   ```
   ENCRYPT PASSWORD sharedsecret {AES128 | AES192 | AES256} ENCRYPTKEY keyname
   ```

   Where:

   - *sharedsecret* is the clear-text shared secret. This value is case-sensitive.
   - `{AES128 | AES192 | AES256}` specifies Advanced Encryption Standard (AES) encryption. Specify one of the values, which represents the desired key length.

- *keyname* is the logical name of the encryption key in the ENCKEYS lookup file.

  Example:

  ```
  ENCRYPT PASSWORD sharedsecret AES256 ENCRYPTKEY mykey1
  ```

5. In the Extract parameter file, use the DBOPTIONS parameter with the DECRYPTPASSWORD option. As input, supply the encrypted shared secret and the Oracle GoldenGate-generated or user-defined decryption key.

   ```
   DBOPTIONS DECRYPTPASSWORD sharedsecret {AES128 | AES192 | AES256} ENCRYPTKEY
   keyname
   ```

   Where:

   - *sharedsecret* is the encrypted shared secret.
   - {AES128 | AES192 | AES256} must be same value that was used for ENCRYPT PASSWORD.
   - *keyname* is the logical name of the encryption key in the ENCKEYS lookup file.

     Example:

     ```
     DBOPTIONS DECRYPTPASSWORD AACAAAAAAAAAAAIALCKDZIRHOJBHOJUH AES256 ENCRYPTKEY
     mykey1
     ```

6. Log in to SQL*Plus as a user with the SYSDBA system privilege.
7. Close and then re-open the wallet.

   ```
   SQL> alter system set encryption wallet close identified by "hsm/
   wallet_password";
   System altered.
   SQL> alter system set encryption wallet open identified by "hsm/
   wallet_password";
   System altered.
   ```

8. Start Extract.

   ```
   START EXTRACT group
   ```

11.2 Using Classic Capture in an Oracle RAC Environment

The following general guidelines apply to Oracle RAC when Extract is operating in classic capture mode.

- During operations, if the primary database instance against which Oracle GoldenGate is running stops or fails for any reason, Extract abends. To resume processing, you can restart the instance or mount the Oracle GoldenGate binaries to another node where the database is running and then restart the Oracle GoldenGate processes. Stop the Manager process on the original node before starting Oracle GoldenGate processes from another node.
- Whenever the number of redo threads changes, the Extract group must be dropped and re-created. For the recommended procedure, see *Administering Oracle GoldenGate for Windows and UNIX*.
- Extract ensures that transactions are written to the trail file in commit order, regardless of the RAC instance where the transaction originated. When Extract is capturing in archived-log-only mode, where one or more RAC instances may be idle, you may need to perform archive log switching on the idle nodes to ensure that operations from the active instances are recorded in the trail file in a timely

manner. You can instruct the Oracle RDBMS to do this log archiving automatically at a preset interval by setting the `archive_lag_target` parameter. For example, to ensure that logs are archived every fifteen minutes, regardless of activity, you can issue the following command in all instances of the RAC system:

```
SQL> alter system set archive_lag_target 900
```

- To process the last transaction in a RAC cluster before shutting down Extract, insert a dummy record into a source table that Oracle GoldenGate is replicating, and then switch log files on all nodes. This updates the Extract checkpoint and confirms that all available archive logs can be read. It also confirms that all transactions in those archive logs are captured and written to the trail in the correct order.

The following table shows some Oracle GoldenGate parameters that are of specific benefit in Oracle RAC.

Table 11-1 Classic Extract Parameters for Oracle RAC

Parameter	Description
`THREADOPTIONS` parameter with the `INQUEUESIZE` and `OUTQUEUESIZE` options	Sets the amount of data that Extract queues in memory before sending it to the target system. Tuning these parameters might increase Extract performance on Oracle RAC.
`TRANLOGOPTIONS` parameter with the `PURGEORPHANEDTRANSACTIONS | NOPURGEORPHANEDTRANSACTIONS` and `TRANSCLEANUPFREQUENCY` options	Controls how Extract handles orphaned transactions, which can occur when a node fails during a transaction and Extract cannot capture the rollback. Although the database performs the rollback on the failover node, the transaction would otherwise remain in the Extract transaction list indefinitely and prevent further checkpointing for the Extract thread that was processing the transaction. By default, Oracle GoldenGate purges these transactions from its list after they are confirmed as orphaned. This functionality can also be controlled on demand with the `SEND EXTRACT` command in GGSCI.

11.3 Mining ASM-stored Logs in Classic Capture Mode

This topic covers additional configuration requirements that apply when Oracle GoldenGate mines transaction logs that are stored in Oracle Automatic Storage Management (ASM).

- Accessing the Transaction Logs in ASM (page 11-8)
- Ensuring ASM Connectivity (page 11-10)

11.3.1 Accessing the Transaction Logs in ASM

Extract must be configured to read logs that are stored in ASM. Depending on the database version, the following options are available:

- Reading Transaction Logs Through the RDBMS (page 11-8)
- ASM Direct Connection (page 11-9)

11.3.1.1 Reading Transaction Logs Through the RDBMS

Use the `TRANLOGOPTIONS` parameter with the `DBLOGREADER` option in the Extract parameter file if the RDBMS is Oracle 11.1.0.7 or Oracle 11.2.0.2 or later 11g R2 versions.

An API is available in those releases (but not in Oracle 11g R1 versions) that uses the database server to access the redo and archive logs. When used, this API enables Extract to use a read buffer size of up to 4 MB in size. A larger buffer may improve the performance of Extract when redo rate is high. You can use the `DBLOGREADERBUFSIZE` option of `TRANLOGOPTIONS` to specify a buffer size.

> **Note:**
>
> `DBLOGREADER` also can be used when the redo and archive logs are on regular disk or on a raw device.

When using `DBLOGREADER` and using Oracle Data Vault, grant the `DV_GOLDENGATE_REDO_ACCESS` Role to the Extract database user in addition to the privileges that are listed in Establishing Oracle GoldenGate Credentials (page 4-1).

11.3.1.2 ASM Direct Connection

If the RDBMS version is not one of those listed in Reading Transaction Logs Through the RDBMS (page 11-8), do the following:

1. Create a user for the Extract process to access the ASM instance directly. Assign this user `SYS` or `SYSDBA` privileges in the ASM instance. Oracle GoldenGate does not support using operating-system authentication for the ASM user. See Table 11-2 (page 11-9) for additional details.

Table 11-2 Extract Database Privileges — ASM Instance

ASM password configuration[1]	Permitted user
ASM instance and the database share a password file	You can use the Oracle GoldenGate source database user if you grant that user `SYSDBA`, or you can use any other database user that has `SYSDBA` privileges.
ASM instance and the source database have separate password files	You can overwrite the ASM password file with the source database password file, understanding that this procedure changes the `SYS` password in the ASM instance to the value that is contained in the database password file, and it also grants ASM access to the other users in the database password file. Save a copy of the ASM file before overwriting it.

[1] To view how the current ASM password file is configured, log on to the ASM instance and issue the following command in SQL*Plus: `SQL> SELECT name, value FROM v$parameter WHERE name = 'remote_login_passwordfile';`

2. Add the ASM user credentials to the Oracle GoldenGate credential store by issuing the `ALTER CREDENTIALSTORE` command. See *Reference for Oracle GoldenGate for Windows and UNIX* for usage instructions and syntax.
3. Specify the ASM login alias in the Extract parameter file by including the `TRANLOGOPTIONS` parameter with the `ASMUSERALIAS` option. For more information about `TRANLOGOPTIONS`, see *Reference for Oracle GoldenGate for Windows and UNIX*.

11.3.2 Ensuring ASM Connectivity

To ensure that the Oracle GoldenGate Extract process can connect to an ASM instance, list the ASM instance in the `tnsnames.ora` file. The recommended method for connecting to an ASM instance when Oracle GoldenGate is running on the database host machine is to use a bequeath (BEQ) protocol. The BEQ protocol does not require a listener. If you prefer to use the TCP/IP protocol, verify that the Oracle listener is listening for new connections to the ASM instance. The `listener.ora` file must contain an entry similar to the following.

```
SID_LIST_LISTENER_ASM =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = ASM)
      (ORACLE_HOME = /u01/app/grid)
    (SID_NAME = +ASM1)
  )
)
```

> **Note:**
>
> A BEQ connection does not work when using a remote Extract configuration. Use `TNSNAMES` with the TCP/IP protocol.

11.4 Ensuring Data Availability for Classic Capture

To ensure the continuity and integrity of capture processing when Extract operates in classic capture mode, enable archive logging. The archive logs provide a secondary data source should the online logs recycle before Extract is finished with them. The archive logs for open transactions must be retained on the system in case Extract needs to recapture data from them to perform a recovery.

> **WARNING:**
>
> If you cannot enable archive logging, there is a high risk that you will need to completely resynchronize the source and target objects and reinstantiate replication should there be a failure that causes an Extract outage while transactions are still active. If you must operate this way, configure the online logs according to the following guidelines to retain enough data for Extract to capture what it needs before the online logs recycle. Allow for Extract backlogs caused by network outages and other external factors, as well as long-running transactions.

In a RAC configuration, Extract must have access to the online and archived logs for all nodes in the cluster, including the one where Oracle GoldenGate is installed.

- Log Retention Requirements per Extract Recovery Mode (page 11-11)
- Log Retention Options (page 11-11)

11.4.1 Log Retention Requirements per Extract Recovery Mode

The following summarizes the different recovery modes that Extract might use and their log-retention requirements:

- By default, the Bounded Recovery mode is in effect, and Extract requires access to the logs only as far back as twice the Bounded Recovery interval that is set with the `BR` parameter. This interval is an integral multiple of the standard Extract checkpoint interval, as controlled by the `CHECKPOINTSECS` parameter. These two parameters control the Oracle GoldenGate Bounded Recovery feature, which ensures that Extract can recover in-memory captured data after a failure, no matter how old the oldest open transaction was at the time of failure. For more information about Bounded Recovery, see *Reference for Oracle GoldenGate for Windows and UNIX*.
- In the unlikely event that the Bounded Recovery mechanism fails when Extract attempts a recovery, Extract reverts to normal recovery mode and must have access to the archived log that contains the beginning of the oldest open transaction in memory at the time of failure and all logs thereafter.

11.4.2 Log Retention Options

Depending on the version of Oracle, there are different options for ensuring that the required logs are retained on the system.

11.4.2.1 Oracle Enterprise Edition 11g and Later

For these versions, Extract can be configured to work with Oracle Recovery Manager (RMAN) to retain the logs that Extract needs for recovery. You enable this feature when you issue the `REGISTER EXTRACT` command. See Creating Process Groups (page 14-1) for more information. To use this feature, the Extract database user must have the following privileges, in addition to the basic privileges listed in Establishing Oracle GoldenGate Credentials (page 4-1).

Table 11-3 Extract Database Privileges —Log Retention

Oracle EE version	Privileges

11.1 and 11.2.0.1	1. Run package to grant Oracle GoldenGate admin privilege.
	`exec dbms_streams_auth.grant_admin_privilege('user')`
	2. Grant the 'become user' privilege.
	`grant become user to user;`

Table 11-3 (Cont.) Extract Database Privileges —Log Retention

Oracle EE version	Privileges
11.2.0.3 and later	Run package to grant Oracle GoldenGate admin privilege. `exec dbms_goldengate_auth.grant_admin_privilege('user')`

When log retention is enabled, Extract retains enough logs to perform a Bounded Recovery, but you can configure Extract to retain enough logs through RMAN for a normal recovery by using the `TRANLOGOPTIONS` parameter with the `LOGRETENTION` option set to `SR`. There also is an option to disable the use of RMAN log retention. Review the options of `LOGRETENTION` in the *Reference for Oracle GoldenGate for Windows and UNIX* before you configure Extract. If you set `LOGRETENTION` to `DISABLED`, see Determining How Much Data to Retain (page 11-12),.

> **Note:**
>
> To support RMAN log retention on Oracle RAC for Oracle versions prior to 11.2.0.3, you must download and install the database patch that is provided in BUGFIX 11879974 before you add the Extract groups.

The RMAN log retention feature creates an underlying (but non-functioning) Oracle Streams Capture process for each Extract group. The name of the Capture is based on the name of the associated Extract group. The log retention feature can operate concurrently with other local Oracle Streams installations. When you create an Extract group, the logs are retained from the current database SCN.

> **Note:**
>
> If the storage area is full, RMAN purges the archive logs even when needed by Extract. This limitation exists so that the requirements of Extract (and other Oracle replication components) do not interfere with the availability of redo to the database.

11.4.2.2 All Other Oracle Versions

For versions of Oracle other than Enterprise Edition, you must manage the log retention process with your preferred administrative tools. Follow the directions in Determining How Much Data to Retain (page 11-12).

11.4.3 Determining How Much Data to Retain

When managing log retention, try to ensure rapid access to the logs that Extract would require to perform a normal recovery (not a Bounded Recovery). See Log Retention Requirements per Extract Recovery Mode (page 11-11). If you must move the archives off the database system, the `TRANLOGOPTIONS` parameter provides a way to specify an alternate location. See Specifying the Archive Location (page 11-13).

The recommended retention period is at least 24 hours worth of transaction data, including both online and archived logs. To determine the oldest log that Extract might need at any given point, issue the `SEND EXTRACT` command with the `SHOWTRANS` option. You might need to do some testing to determine the best retention time given your data volume and business requirements.

If data that Extract needs during processing was not retained, either in online or archived logs, one of the following corrective actions might be required:

- Alter Extract to capture from a later point in time for which log data is available (and accept possible data loss on the target).
- Resynchronize the source and target data, and then start the Oracle GoldenGate environment over again.

11.4.4 Purging Log Archives

Make certain not to use backup or archive options that cause old archive files to be overwritten by new backups. Ideally, new backups should be separate files with different names from older ones. This ensures that if Extract looks for a particular log, it will still exist, and it also ensures that the data is available in case it is needed for a support case.

11.4.5 Specifying the Archive Location

If the archived logs reside somewhere other than the Oracle default directory, specify that directory with the `ALTARCHIVELOGDEST` option of the `TRANLOGOPTIONS` parameter in the Extract parameter file.

You might also need to use the `ALTARCHIVEDLOGFORMAT` option of `TRANLOGOPTIONS` if the format that is specified with the Oracle parameter `LOG_ARCHIVE_FORMAT` contains sub-directories. `ALTARCHIVEDLOGFORMAT` specifies an alternate format that removes the sub-directory from the path. For example, `%T/log_%t_%s_%r.arc` would be changed to `log_%t_%s_%r.arc`. As an alternative to using `ALTARCHIVEDLOGFORMAT`, you can create the sub-directory manually, and then move the log files to it.

11.4.6 Mounting Logs That are Stored on Other Platforms

If the online and archived redo logs are stored on a different platform from the one that Extract is built for, do the following:

- NFS-mount the archive files.
- Map the file structure to the structure of the source system by using the `LOGSOURCE` and `PATHMAP` options of the Extract parameter `TRANLOGOPTIONS`. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.

11.5 Configuring Classic Capture in Archived Log Only Mode

You can configure Extract to read exclusively from the archived logs. This is known as **Archived Log Only (ALO)** mode. In this mode, Extract reads exclusively from archived logs that are stored in a specified location. ALO mode enables Extract to use production logs that are shipped to a secondary database (such as a standby) as the

data source. The online logs are not used at all. Oracle GoldenGate connects to the secondary database to get metadata and other required data as needed. As an alternative, ALO mode is supported on the production system.

> **Note:**
>
> ALO mode is not compatible with Extract operating in integrated capture mode.

- Limitations and Requirements for Using ALO Mode (page 11-14)
- Configuring Extract for ALO mode (page 11-14)

11.5.1 Limitations and Requirements for Using ALO Mode

Observe the following limitations and requirements when using Extract in ALO mode.

- Log resets (`RESETLOG`) cannot be done on the source database after the standby database is created.
- ALO cannot be used on a standby database if the production system is Oracle RAC and the standby database is non-RAC. In addition to both systems being Oracle RAC, the number of nodes on each system must be identical.
- ALO on Oracle RAC requires a dedicated connection to the source server. If that connection is lost, Oracle GoldenGate processing will stop.
- It is a best practice to use separate archive log directories when using Oracle GoldenGate for Oracle RAC in ALO mode. This will avoid any possibility of the same file name showing up twice, which could result in Extract returning an "out of order scn" error.
- The `LOGRETENTION` parameter defaults to `DISABLED` when Extract is in ALO mode. You can override this with a specific `LOGRETENTION` setting, if needed.

11.5.2 Configuring Extract for ALO mode

To configure Extract for ALO mode, follow these steps as part of the overall process for configuring Oracle GoldenGate, as documented in Configuring Capture in Classic Mode (page 8-1).

1. Enable supplemental logging at the table level and the database level for the tables in the source database. (See Configuring Logging Properties (page 3-2).)
2. When Oracle GoldenGate is running on a different server from the source database, make certain that SQL*Net is configured properly to connect to a remote server, such as providing the correct entries in a `TNSNAMES` file. Extract must have permission to maintain a SQL*Net connection to the source database.
3. Use a SQL*Net connect string for the name of the user in the credential store that is assigned to the process. Specify the alias of this user in the following:
   - The `USERIDALIAS` parameter in the parameter file of every Oracle GoldenGate process that connects to that database.
   - The `USERIDALIAS` portion of the `DBLOGIN` command in GGSCI.

> **Note:**
>
> If you have a standby server that is local to the server that Oracle GoldenGate is running on, you do not need to use a connect string for the user specified in `USERIDALIAS`. You can just supply the user login name.

See *Administering Oracle GoldenGate for Windows and UNIX* for more information about using a credential store.

4. Use the Extract parameter `TRANLOGOPTIONS` with the `ARCHIVEDLOGONLY` option. This option forces Extract to operate in ALO mode against a primary or logical standby database, as determined by a value of `PRIMARY` or `LOGICAL STANDBY` in the `db_role` column of the `v$database` view. The default is to read the online logs. `TRANLOGOPTIONS` with `ARCHIVEDLOGONLY` is not needed if using ALO mode against a physical standby database, as determined by a value of `PHYSICAL STANDBY` in the `db_role` column of `v$database`. Extract automatically operates in ALO mode if it detects that the database is a physical standby.
5. Other `TRANLOGOPTIONS` options might be required for your environment. For example, depending on the copy program that you use, you might need to use the `COMPLETEARCHIVEDLOGONLY` option to prevent Extract errors.
6. Use the `MAP` parameter for Extract to map the table names to the source object IDs. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.
7. Add the Extract group by issuing the `ADD EXTRACT` command with a timestamp as the `BEGIN` option, or by using `ADD EXTRACT` with the `SEQNO` and `RBA` options. It is best to give Extract a known start point at which to begin extracting data, rather than by using the `NOW` argument. The start time of `NOW` corresponds to the time of the current *online* redo log, but an ALO Extract cannot read the online logs, so it must wait for that log to be archived when Oracle switches logs. The timing of the switch depends on the size of the redo logs and the volume of database activity, so there might be a lag between when you start Extract and when data starts being captured. This can happen in both regular and RAC database configurations.

11.6 Configuring Classic Capture in Oracle Active Data Guard Only Mode

You can configure Classic Extract to access both redo data and metadata in real-time to successfully replicate source database activities using Oracle Active Data Guard. This is known as *Active Data Guard* (ADG) mode. ADG mode enables Extract to use production logs that are shipped to a standby database as the data source. The online logs are not used at all. Oracle GoldenGate connects to the standby database to get metadata and other required data as needed.

This mode is useful in load sensitive environments where ADG is already in place or can be implemented. It can also be used as cost effective method to implement high availability using the ADG Broker role planned (switchover) and failover (unplanned) changes. In an ADG configuration, switchover and failover are considered *roles*. When either of the operations occur, it is considered a *role change*. For more information, see Oracle Data Guard Concepts and Administration and Oracle Data Guard Broker.

You can configure Integrated Extract to fetch table data and metadata required for the fetch from an ADG instead of the source database. This is possible because an ADG is a physical replica of the source database. Fetching from an ADG using the `FETCHUSER` parameter is supported by Extract in all configurations except when running as Classic Extract. Classic Extract already has the ability to connect directly to an ADG and mine its redo logs and fetch from it using standard connection information supplied using the `USERID` parameter. The impact to the source database is minimized because Extract gathers information from the source database at startup, including compatibility level, database type, and source database validation checks, when fetching from an ADG.

All previous fetch functionality and parameters are supported.

> **Note:**
>
> Integrated Extract cannot capture from a standby database because it requires `READ` and `WRITE` access to the database, and an ADG standby only provides `READ ONLY` access.

- Limitations and Requirements for Using ADG Mode (page 11-16)
- Configuring Classic Extract for ADG Mode (page 11-17)
- Migrating Classic Extract To and From an ADG Database (page 11-18)
- Handling Role Changes In an ADG Configuration (page 11-18)

11.6.1 Limitations and Requirements for Using ADG Mode

Observe the following limitations and requirements when using Extract in ADG mode.

- Extract in ADG mode will only apply redo data that has been applied to the standby database by the apply process. If Extract runs ahead of the standby database, it will wait for the standby database to catch up.
- You must explicitly specify ADG mode in your classic Extract parameter file to run extract on the standby database.
- You must specify the database user and password to connect to the ADG system because fetch and other metadata resolution occurs in the database.
- The number of redo threads in the standby logs in the standby database must match the number of nodes from the primary database.
- No new RAC instance can be added to the primary database after classic Extract has been created on the standby database. If you do add new instances, the redo data from the new thread will not be captured by classic Extract.
- Archived logs and standby redo logs accessed from the standby database will be an exact duplicate of the primary database. The size and the contents will match, including redo data, transactional data, and supplemental data. This is guaranteed by a properly configured ADG deployment.
- ADG role changes are infrequent and require user intervention in both cases.
- With a switchover, there will be an indicator in the redo log file header (end of the redo log or EOR marker) to indicate end of log stream so that classic Extract on

the standby can complete the RAC coordination successfully and ship all of the committed transactions to the trail file.

- With a failover, a new incarnation is created on both the primary and the standby databases with a new incarnation ID, `RESETLOG` sequence number, and SCN value.
- You must connect to the primary database from GGSCI to add `TRANDATA` or `SCHEMATRANDATA` because this is done on the primary database.
- DDL triggers cannot be used on the standby database, in order to support DDL replication (except `ADDTRANDATA`). You must install the Oracle GoldenGate DDL package on the primary database.
- DDL `ADDTRANDATA` is not supported in ADG mode; you must use `ADDSCHEMATRANDATA` for DDL replication.
- When adding extract on the standby database, you must specify the starting position using a specific SCN value, timestamp, or log position. Relative timestamp values, such as `NOW`, become ambiguous and may lead to data inconsistency.
- When adding extract on the standby database, you must specify the number of threads that will include all of the relevant threads from the primary database.
- During or after failover or switchover, no thread can be added or dropped from either primary or standby databases.
- Classic Extract will only use one intervening `RESETLOG` operation.
- If you do not want to relocate your Oracle GoldenGate installation, then you must position it in a shared space where the Oracle GoldenGate installation directory can be accessed from both the primary and standby databases.
- If you are moving capture off of an ADG standby database to a primary database, then you must point your net alias to the primary database and you must remove the `TRANLOG` options.
- Only Oracle Database releases that are running with compatibility setting of 10.2 or higher (10g Release 2) are supported.
- Classic Extract cannot use the `DBLOGREADER` option. Use `ASMUSER` (there is approximately a 20gb/hr read limit) or move the online and archive logs outside of the Application Security Manager on both the primary and the standby databases.

11.6.2 Configuring Classic Extract for ADG Mode

To configure Classic Extract for ADG mode, follow these steps as part of the overall process for configuring Oracle GoldenGate, as documented in Configuring Capture in Classic Mode (page 8-1).

1. Enable supplemental logging at the table level and the database level for the tables in the *primary* database using the `ADD SCHEMATRANDATA` parameter. If necessary, create a DDL capture. (See Configuring Logging Properties (page 3-2).)
2. When Oracle GoldenGate is running on a different server from the source database, make certain that SQL*Net is configured properly to connect to a remote server, such as providing the correct entries in a `TNSNAMES` file. Extract must have permission to maintain a SQL*Net connection to the source database.
3. On the *standby* database, use the Extract parameter `TRANLOGOPTIONS` with the `MINEFROMACTIVEDG` option. This option forces Extract to operate in ADG mode

against a standby database, as determined by a value of `PRIMARY` or `LOGICAL STANDBY` in the `db_role` column of the `v$database` view.

Other `TRANLOGOPTIONS` options might be required for your environment. For example, depending on the copy program that you use, you might need to use the `COMPLETEARCHIVEDLOGONLY` option to prevent Extract errors.

4. On the *standby* database, add the Extract group by issuing the `ADD EXTRACT` command specifying the number of threads active on the *primary* database at the given SCN. The timing of the switch depends on the size of the redo logs and the volume of database activity, so there might be a limited lag between when you start Extract and when data starts being captured. This can happen in both regular and RAC database configurations.

11.6.3 Migrating Classic Extract To and From an ADG Database

You must have your parameter files, checkpoint files, bounded recovery files, and trail files stored in shared storage or copied to the ADG database before attempting to migrate a classic Extract to or from an ADG database. Additionally, you must ensure that there has not been any intervening role change or Extract will mine the same branch of redo.

Use the following steps to move to an ADG database:

1. Edit the parameter file `ext1.prm` to add the following parameters:

   ```
   DBLOGIN USERID userid@ADG PASSWORD password
   TRANLOGOPTIONS MINEFROMACTIVEDG
   ```

2. Start Extract by issuing the `START EXTRACT ext1` command.

Use the following steps to move from an ADG database:

1. Edit the parameter file `ext1.prm` to remove the following parameters:

   ```
   DBLOGIN USERID userid@ADG PASSWORD password
   TRANLOGOPTIONS MINEFROMACTIVEDG
   ```

2. Start Extract by issuing the `START EXTRACT ext1` command.

11.6.4 Handling Role Changes In an ADG Configuration

In a role change involving a standby database, all sessions in the primary and the standby database are first disconnected including the connections used by Extract. Then both databases are shut down, then the original primary is mounted as a standby database, and the original standby is opened as the primary database.

The procedure for a role change is determined by the initial deployment of Classic Extract and the deployment relation that you want, database or role. The following table outlines the four possible role changes and is predicated on an ADG configuration comprised of two databases, `prisys` and `stansys`. The `prisys` system contains the primary database and the `stansys` system contains the standby database; `prisys` has two redo threads active, whereas `stansys` has four redo threads active.

Initial Deployment Primary (`prisys`)	Initial Deployment ADG (`stansys`)
Original Deployment:	

Initial Deployment Primary (prisys)	Initial Deployment ADG (stansys)
ext1.prm DBLOGIN USERID *userid*@prisys, PASSWORD *password*	ext1.prm DBLOGIN USERID *userid*@stansys, PASSWORD *password* TRANLOGOPTIONS MINEFROMACTIVEDG
Database Related:	
After Role Transition: Classic Extract to ADG	After Role Transition: ADG to classic Extract
1. Edit the ext1.prm file to add: TRANLOGOPTIONS MINEFROMACTIVEDG	1. Edit ext1.prm and remove: TRANLOGOPTIONS MINEFROMACTIVEDG
2. If a failover, add TRANLOGOPTIONS USEPREVRESETLOGSID.	2. If a failover, add TRANLOGOPTIONS USEPREVRESETLOGSID.
3. Start Extract: START EXTRACT ext1 Extract will abend once it reaches the role transition point, then it does an internal BR_RESET and moves both the I/O checkpoint and current checkpoint to SCN *s*.	3. Start Extract: START EXTRACT ext1 Extract will abend once it reaches the role transition point, then it does an internal BR_RESET and moves both the I/O checkpoint and current checkpoint to SCN *s*.
4. If failover, edit the parameter file again and remove: TRANLOGOPTIONS USEPREVRESETLOGSID	4. If failover, edit the parameter file again and remove: TRANLOGOPTIONS USEPREVRESETLOGSID
5. Execute ALTER EXTRACT ext1 SCN #, where # is the SCN value from role switch message.	5. Execute ALTER EXTRACT ext1 SCN #, where # is the SCN value from role switch message.
6. Based on the thread counts, do one of the following: If the thread counts are same between the databases, then execute the START EXTRACT ext1; command. or If thread counts are different between the databases, then execute the following commands: DROP EXTRACT ext1 ADD EXTRACT ext1 THREADS t BEGIN SCN s START EXTRACT ext1	6. Based on the thread counts, do one of the following: If the thread counts are same between the databases, then execute the START EXTRACT ext1; command. or If thread counts are different between the databases, then execute the following commands: DROP EXTRACT ext1 ADD EXTRACT ext1 THREADS t BEGIN SCN s START EXTRACT ext1
Role Related:	

Initial Deployment Primary (prisys)	Initial Deployment ADG (stansys)
After Role Transition: Classic Extract to classic Extract 1. Edit ext1.prm to change the database system to the standby system: DBLOGIN USERID userid@stansys, PASSWORD *password* 2. If a failover, add TRANLOGOPTIONS USEPREVRESETLOGSID. 3. Start Extract: START EXTRACT ext1 Extract will abend once it reaches the role transition point, then it does an internal BR_RESET and moves both the I/O checkpoint and current checkpoint to SCN s. 4. If failover, edit the parameter file again and remove: TRANLOGOPTIONS USEPREVRESETLOGSID 5. Execute ALTER EXTRACT ext1 SCN #, where# is the SCN value from role switch message. 6. Based on the thread counts, do one of the following: If the thread counts are same between the databases, then execute the START EXTRACT ext1; command. or If thread counts are different between the databases, then execute the following commands: DROP EXTRACT ext1 ADD EXTRACT ext1 THREADS t BEGIN SCN s START EXTRACT ext1	After Role Transition: ADG to ADG 1. Edit ext1.prm to change the database system to the primary system: DBLOGIN USERID userid@prisys, PASSWORD *password* 2. If a failover, add TRANLOGOPTIONS USEPREVRESETLOGSID. 3. Start Extract: START EXTRACT ext1 Extract will abend once it reaches the role transition point, then it does an internal BR_RESET and moves both the I/O checkpoint and current checkpoint to SCN s. 4. If failover, edit the parameter file again and remove: TRANLOGOPTIONS USEPREVRESETLOGSID 5. Execute ALTER EXTRACT ext1 SCN #, where# is the SCN value from role switch message. 6. Based on the thread counts, do one of the following: If the thread counts are same between the databases, then execute the START EXTRACT ext1; command. or If thread counts are different between the databases, then execute the following commands: DROP EXTRACT ext1 ADD EXTRACT ext1 THREADS t BEGIN SCN s START EXTRACT ext1

11.7 Avoiding Log-read Bottlenecks in Classic Capture

When Oracle GoldenGate captures data from the redo logs, I/O bottlenecks can occur because Extract is reading the same files that are being written by the database logging mechanism. Performance degradation increases with the number of Extract processes that read the same logs. You can:

- Try using faster drives and a faster controller. Both Extract and the database logging mechanism will be faster on a faster I/O system.
- Store the logs on RAID 0+1. Avoid RAID 5, which performs checksums on every block written and is not a good choice for high levels of continuous I/O.

12

Additional Configuration Steps For Using Nonintegrated Replicat

This chapter contains instructions that are specific only to Replicat when operating in *nonintegrated* mode. When Replicat operates in nonintegrated mode, triggers, cascade constraints, and unique identifiers must be properly configured in an Oracle GoldenGate environment.
This chapter is a supplement to the basic configuration requirements that are documented in Configuring Oracle GoldenGate Apply (page 9-1).

Topics:

12.1 Disabling Triggers and Referential Cascade Constraints on Target Tables

Triggers and cascade constraints must be disabled on Oracle target tables when Replicat is in nonintegrated mode. Oracle GoldenGate provides some options to handle triggers or cascade constraints automatically, depending on the Oracle version:

- For Oracle 11.2.0.2 and later 11gR2 versions, Replicat automatically disables the work performed by triggers during its session. It does not disable a trigger, but instead prevents the trigger body from executing. The `WHEN` portion of the trigger must still compile and execute correctly to avoid database errors. To enable triggers to fire, or to disable them manually, use the `NOSUPPRESSTRIGGERS` option of `DBOPTIONS` and place the statement after the `USERIDALIAS` parameter. To allow a specific trigger to fire, you can use the following database procedure, where *`trigger_owner`* is the owner of the trigger and *`trigger_name`* is the name of the trigger. Once the procedure is called with `FALSE` for a particular trigger, it remains set until the procedure is called with `TRUE`.
- `dbms_ddl.set_trigger_firing_property(`*`trigger_owner`* `"`*`trigger_name`*`", FALSE)`
- For Oracle 11.2.0.2 and later 11gR2 versions, you can use the `DBOPTIONS` parameter with the `DEFERREFCONST` option to delay the checking and enforcement of cascade update and cascade delete constraints until the Replicat transaction commits.
- For other Oracle versions, you must disable triggers and integrity constraints or alter them manually to ignore the Replicat database user.

Constraints must be disabled in nonintegrated Replicat mode because Oracle GoldenGate replicates DML that results from the firing of a trigger or a cascade constraint. If the same trigger or constraint gets activated on the target table, it becomes redundant because of the replicated version, and the database returns an

error. Consider the following example, where the source tables are `emp_src` and `salary_src` and the target tables are `emp_targ` and `salary_targ`.

1. A delete is issued for `emp_src`.
2. It cascades a delete to `salary_src`.
3. Oracle GoldenGate sends both deletes to the target.
4. The parent delete arrives first and is applied to `emp_targ`.
5. The parent delete cascades a delete to `salary_targ`.
6. The cascaded delete from `salary_src` is applied to `salary_targ`.
7. The row cannot be located because it was already deleted in step 5.

12.2 Deferring Constraint Checking on Target Tables

When Replicat is in nonintegrated mode, you may need to defer constraint checking on the target.

1. If constraints are `DEFERRABLE` on the source, the constraints on the target must also be `DEFERRABLE`. You can use one of the following parameter statements to defer constraint checking until a Replicat transaction commits:
   - Use `SQLEXEC` at the root level of the Replicat parameter file to defer the constraints for an entire Replicat session.

     ```
     SQLEXEC ("alter session set constraint deferred")
     ```

   - Use the Replicat parameter `DBOPTIONS` with the `DEFERREFCONST` option to delay constraint checking for each Replicat transaction.
2. You might need to configure Replicat to overcome integrity errors caused by transient primary-key duplicates. Transient primary-key duplicates are duplicates that occur temporarily during the execution of a transaction, but are resolved by transaction commit time. This kind of operation typically uses a `SET x = x+n` formula or some other manipulation that shifts values so that a new value equals an existing one.

   The following illustrates a sequence of value changes that can cause a transient primary-key duplicate if constraints are not deferred. The example assumes the primary key column is `CODE` and the current key values (before the updates) are 1, 2, and 3.

   ```
   update item set code = 2 where code = 1;
   update item set code = 3 where code = 2;
   update item set code = 4 where code = 3;
   ```

In this example, when Replicat applies the first update to the target, there is an ORA-00001 (unique constraint) error because the key value of 2 already exists in the table. The Replicat transaction returns constraint violation errors. By default, Replicat does not handle these violations and abends.

- Handling Transient Primary-key Duplicates in Versions Earlier than 11.2.0.2 (page 12-3)
- Handling Transient Primary-key Duplicates in Version 11.2.0.2 or Later (page 12-3)

12.2.1 Handling Transient Primary-key Duplicates in Versions Earlier than 11.2.0.2

To handle transient primary-key duplicates in versions earlier than 11.2.0.2, use the Replicat parameter `HANDLETPKUPDATE`. In this configuration, a nonintegrated Replicat handles transient primary-key updates by temporarily deferring constraints. To support this functionality, you must create or alter the constraints as `DEFERRABLE INITIALLY IMMEDIATE` on the target tables. If the constraints are not `DEFERRABLE`, Replicat handles the errors according to rules that are specified with the `HANDLECOLLISIONS` and `REPERROR` parameters, if they exist, or else it abends.

12.2.2 Handling Transient Primary-key Duplicates in Version 11.2.0.2 or Later

For versions later than 11.2.0.2, a nonintegrated Replicat by default tries to resolve transient primary-key duplicates automatically by using a workspace in Oracle Workspace Manager. In this configuration, Replicat can defer the constraint checking until commit time without requiring the constraints to be explicitly defined as deferrable.

The requirements for automatic handling of transient primary-key duplicates are:

- Grant the Replicat database user access to the following Oracle function:

  `DBMS_XSTREAM_GG.ENABLE_TDUP_WORKSPACE()`

- The target tables cannot have deferrable constraints; otherwise Replicat returns an error and abends.

To handle tables with deferrable constraints, make certain the constraints are `DEFERRABLE INITIALLY IMMEDIATE`, and use the `HANDLETPKUPDATE` parameter in the `MAP` statement that maps that table. The `HANDLETPKUPDATE` parameter overrides the default of handling the duplicates automatically.The use of a workspace affects the following Oracle GoldenGate error handling parameters:

- `HANDLECOLLISIONS`
- `REPERROR`

When Replicat enables a workspace in Oracle Workspace Manager, it ignores the error handling that is specified by Oracle GoldenGate parameters such as `HANDLECOLLISIONS` and `REPERROR`. Instead, Replicat aborts its grouped transaction (if `BATCHSQL` is enabled), and then retries the update in normal mode by using the active workspace. If ORA-00001 occurs again, Replicat rolls back the transaction and then retries the transaction with error-handling rules in effect again.

> **Note:**
>
> If Replicat encounters ORA-00001 for a non-update record, the error-handling parameters such as `HANDLECOLLISIONS` and `REPERROR` handle it.

A workspace cannot be used if the operation that contains a transient primary-key duplicate also has updates to any out-of-line columns, such as LOB and XMLType. Therefore, these cases are not supported, and any such cases can result in undetected data corruption on the target. An example of this is:

```
update T set PK = PK + 1, C_LOB = 'ABC';
```

13
Configuring DDL Support

This chapter contains information to help you understand and configure DDL support in Oracle GoldenGate.

Topics:

13.1 Prerequisites for Configuring DDL

Extract can capture DDL operations from a source Oracle Database through the use of a special DDL trigger or natively through the Oracle logmining server. Which of these methods you can use depends on the Extract capture mode and the version of the source Oracle Database. This section describes the available support in each capture mode. See Choosing Capture and Apply Modes (page 5-1) for more information about these modes.

13.1.1 Support for DDL capture in integrated capture mode

The integrated capture mode of Extract supports two DDL capture methods:

- **Oracle 11.2.0.4 or later**: Oracle Databases that have the database `COMPATIBLE` parameter set to 11.2.0.4 or higher support DDL capture through the database logmining server. This method is known as *native DDL capture* (also known as *triggerless DDL capture*). No trigger or installed supportive objects are required. Native DDL capture is the only supported method for capturing DDL from a multitenant container database. For downstream mining, the source database must also have database `COMPATIBLE` set to 11.2.0.4 or higher to support DDL capture through the database logmining server.
- **Versions earlier than 11.2.0.4**: Oracle Databases that have the `COMPATIBLE` parameter set to anything earlier than 11.2.0.4 require the use of the Oracle GoldenGate DDL trigger. To use trigger-based DDL capture, you must install the DDL trigger and supporting database objects before you configure Extract for DDL support. See Installing Trigger-Based DDL Capture (page D-1) for instructions.

13.1.2 Support for DDL capture in classic capture mode

Classic capture mode requires the use of the Oracle GoldenGate DDL trigger to capture DDL from an Oracle Database. Native DDL capture is not supported by classic capture mode. DDL capture from a multitenant container database is not supported by classic capture mode.

To use trigger-based DDL capture, you must install the DDL trigger and supporting database objects before you configure Extract for DDL support. See Installing Trigger-Based DDL Capture (page D-1) for instructions.

13.2 Overview of DDL Synchronization

Oracle GoldenGate supports the synchronization of DDL operations from one database to another. DDL synchronization can be active when:

- business applications are actively accessing and updating the source and target objects.
- Oracle GoldenGate transactional data synchronization is active.

The components that support the replication of DDL and the replication of transactional data changes (DML) are independent of each other. Therefore, you can synchronize:

- just DDL changes
- just DML changes
- both DDL and DML

For a list of supported objects and operations for DDL support for Oracle, see Supported Objects and Operations in Oracle DDL (page 1-26).

13.3 Limitations of Oracle GoldenGate DDL Support

This topic contains some limitations of the DDL feature. For any additional limitations that were found after this documentation was published, see the *Release Notes for Oracle GoldenGate for Windows and UNIX*.

13.3.1 DDL Statement Length

Oracle GoldenGate measures the length of a DDL statement in bytes, not in characters. The supported length is approximately 4 MB, allowing for some internal overhead that can vary in size depending on the name of the affected object and its DDL type, among other characteristics. If the DDL is longer than the supported size, Extract will issue a warning and ignore the DDL operation.

If Extract is capturing DDL by means of the DDL trigger, the ignored DDL is saved in the marker table. You can capture Oracle DDL statements that are ignored, as well as any other Oracle DDL statement, by using the `ddl_ddl2file.sql` script, which saves the DDL operation to a text file in the `USER_DUMP_DEST` directory of Oracle. The script prompts for the following input:

- The name of the schema that contains the Oracle GoldenGate DDL objects, which is specified in the `GLOBALS` file.
- The Oracle GoldenGate marker sequence number, which is recorded in the Extract report file when `DDLOPTIONS` with the `REPORT` option is used in the Extract parameter file.
- A name for the output file.

13.3.2 Supported Topologies

Oracle GoldenGate supports DDL synchronization only in a like-to-like configuration. The source and target object definitions must be identical.

Oracle GoldenGate does not support DDL on a standby database.

Oracle GoldenGate supports DDL replication in all supported unidirectional configurations, and in bidirectional configurations between two, and only two, systems. For special considerations in an Oracle active-active configuration, see Propagating DDL in Active-Active (Bidirectional) Configurations (page 13-26).

13.3.3 Filtering, Mapping, and Transformation

DDL operations cannot be transformed by any Oracle GoldenGate process. However, source DDL can be mapped and filtered to a different target object by a primary Extract or a Replicat process. Mapping or filtering of DDL by a data-pump Extract is not permitted, and the DDL is passed as it was received from the primary Extract.

For example, `ALTER TABLE TableA` is processed by a data pump as `ALTER TABLE TableA`. It cannot be mapped by that process as `ALTER TABLE TableB`, regardless of any `TABLE` statements that specify otherwise.

13.3.4 Renames

`RENAME` operations on tables are converted to the equivalent `ALTER TABLE RENAME` so that a schema name can be included in the target DDL statement. For example `RENAME tab1 TO tab2` could be changed to `ALTER TABLE schema.tab1 RENAME TO schema.tab2`. The conversion is reported in the Replicat process report file.

13.3.5 Interactions Between Fetches from a Table and DDL

Oracle GoldenGate supports some data types by identifying the modified row from the redo stream and then querying the underlying table to fetch the changed columns. For instance, in classic capture, partial updates on LOBs (modifications done via `dbms_lob package`) are supported by identifying the modified row and the LOB column from the redo log, and then querying for the LOB column value for the row from the base table. A similar technique is employed to support UDT (both in classic and integrated capture).

> **Note:**
>
> Integrated capture only requires fetch for UDT when *not* using native object support.

Such fetch-based support is implemented by issuing a flashback query to the database based on the SCN (System Change Number) at which the transaction committed. The flashback query feature has certain limitations. Certain DDL operations act as barriers such that flashback queries to get data prior to these DDLs do not succeed. Examples of such DDL are `ALTER TABLE MODIFY COLUMN` and `ALTER TABLE DROP COLUMN`.

Thus, in cases where there is Extract capture lag, an intervening DDL may cause fetch requests for data prior to the DDL to fail. In such cases, Extract falls back and fetches the current snapshot of the data for the modified column. There are several limitations to this approach: First, the DDL could have modified the column that Extract needs to fetch (for example, suppose the intervening DDL added a new attribute to the UDT that is being captured). Second, the DDL could have modified one of the columns that Extract uses as a logical row identifier. Third, the table could have been renamed before Extract had a chance to fetch the data.

To prevent fetch-related inconsistencies such as these, take the following precautions while modifying columns.

1. Pause all DML to the table.
2. Wait for Extract to finish capturing all remaining redo, and wait for Replicat to finish processing the captured data from trail. To determine whether Replicat is finished, issue the following command in GGSCI until you see a message that there is no more data to process.

   ```
   INFO REPLICAT group
   ```
3. Execute the DDL on the source.
4. Resume source DML operations.

13.3.6 Comments in SQL

If a source DDL statement contains a comment in the middle of an object name, that comment will appear at the end of the object name in the target DDL statement. For example:

Source:

```
CREATE TABLE hr./*comment*/emp ...
```

Target:

```
CREATE TABLE hr.emp /*comment*/ ...
```

This does not affect the integrity of DDL synchronization. Comments in any other area of a DDL statement remain in place when replicated.

13.3.7 Compilation Errors

If a `CREATE` operation on a trigger, procedure, function, or package results in compilation errors, Oracle GoldenGate executes the DDL operation on the target anyway. Technically, the DDL operations themselves completed successfully and should be propagated to allow dependencies to be executed on the target, for example in recursive procedures.

13.3.8 Interval Partitioning

DDL replication is unaffected by interval partitioning, because the DDL is implicit. However, this is system generated name so Replicat cannot convert this to the target.I believe this is expected behavior. You must drop the partition on the source. For example:

```
alter table t2 drop partition for (20);
```

13.3.9 DML or DDL Performed Inside a DDL Trigger

DML or DDL operations performed from within a DDL trigger are not captured.

13.3.10 LogMiner Data Dictionary Maintenance

Oracle recommends that you gather dictionary statistics *after* the Extract is registered (logminer session) and the logminer dictionary is loaded, or after any significant DDL activity on the database.

13.4 Configuration Guidelines for DDL Support

The following are guidelines to take into account when configuring Oracle GoldenGate processes to support DDL replication.

13.4.1 Database Privileges

See Establishing Oracle GoldenGate Credentials (page 4-1) for database privileges that are required for Oracle GoldenGate to support DDL capture and replication.

13.4.2 Parallel Processing

If using parallel Extract and/or Replicat processes, keep related DDL and DML together in the same process stream to ensure data integrity. Configure the processes so that:

- all DDL and DML for any given object are processed by the same Extract group and by the same Replicat group.
- all objects that are relational to one another are processed by the same process group.

For example, if `ReplicatA` processes DML for `Table1`, then it should also process the DDL for `Table1`. If `Table2` has a foreign key to `Table1`, then its DML and DDL operations also should be processed by `ReplicatA`.

If an Extract group writes to multiple trails that are read by different Replicat groups, Extract sends all of the DDL to all of the trails. Use each Replicat group to filter the DDL by using the filter options of the `DDL` parameter in the Replicat parameter file.

13.4.3 Object Names

Oracle GoldenGate preserves the database-defined object name, case, and character set. This support preserves single-byte and multibyte names, symbols, and accent characters at all levels of the database hierarchy.

Object names must be fully qualified with their two-part or three-part names when supplied as input to any parameters that support DDL synchronization. You can use the question mark (**?**) and asterisk (*****) wildcards to specify object names in configuration parameters that support DDL synchronization, but the wildcard specification also must be fully qualified as a two-part or three-part name. For more information about support for wildcards, see *Administering Oracle GoldenGate for Windows and UNIX*. To process wildcards correctly, the `WILDCARDRESOLVE` parameter is

set to DYNAMIC by default. If WILDCARDRESOLVE is set to anything else, the Oracle GoldenGate process that is processing DDL operations will abend and write the error to the process report.

13.4.4 Data Definitions

Because DDL support requires a like-to-like configuration, the ASSUMETARGETDEFS parameter must be used in the Replicat parameter file. Replicat will abend if objects are configured for DDL support and the SOURCEDEFS parameter is being used. For more information about ASSUMETARGETDEFS, see *Reference for Oracle GoldenGate for Windows and UNIX*.

For more information about using a definitions file, see *Administering Oracle GoldenGate for Windows and UNIX*.

13.4.5 Truncates

TRUNCATE statements can be supported as follows:

- As part of the Oracle GoldenGate full DDL support, which supports TRUNCATE TABLE, ALTER TABLE TRUNCATE PARTITION, and other DDL. This is controlled by the DDL parameter (see Enabling DDL Support (page 13-12).)
- As standalone TRUNCATE support. This support enables you to replicate TRUNCATE TABLE, but no other DDL. The GETTRUNCATES parameter controls the standalone TRUNCATE feature. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.

To avoid errors from duplicate operations, only one of these features can be active at the same time.

13.4.6 Initial Synchronization

To configure DDL replication, start with a target database that is synchronized with the source database. DDL support is compatible with the Replicat initial load method.

Before executing an initial load, disable DDL extraction and replication. DDL processing is controlled by the DDL parameter in the Extract and Replicat parameter files.

After initial synchronization of the source and target data, use all of the source sequence values at least once with NEXTVAL before you run the source applications. You can use a script that selects NEXTVAL from every sequence in the system. This must be done while Extract is running.

13.4.7 Data Continuity After CREATE or RENAME

To replicate DML operations on new Oracle tables resulting from a CREATE or RENAME operation, the names of the new tables must be specified in TABLE and MAP statements in the parameter files. You can use wildcards to make certain that they are included.

To create a new user with CREATE USER and then move new or renamed tables into that schema, the new user name must be specified in TABLE and MAP statements. To create a new user fin2 and move new or renamed tables into that schema, the parameter statements could look as follows, depending on whether you want the fin2 objects mapped to the same, or different, schema on the target:

Extract:

```
TABLE fin2.*;
```

Replicat:

```
MAP fin2.*, TARGET different_schema.*;
```

13.5 Understanding DDL Scopes

Database objects are classified into scopes. A scope is a category that defines how DDL operations on an object are handled by Oracle GoldenGate. The scopes are:

- MAPPED
- UNMAPPED
- OTHER

The use of scopes enables granular control over the filtering of DDL operations, string substitutions, and error handling.

- Mapped Scope (page 13-8)
- Unmapped Scope (page 13-10)
- Other Scope (page 13-11)

13.5.1 Mapped Scope

Objects that are specified in TABLE and MAP statements are of MAPPED scope. Extraction and replication instructions in those statements apply to both data (DML) and DDL on the specified objects, unless override rules are applied.

For objects in TABLE and MAP statements, the DDL operations listed in the following table are supported.

Table 13-1 Objects That Can be Mapped in MAP and TABLE Statements

Operations	On any of these Objects[1]
CREATE	TABLE[3]
ALTER	INDEX
DROP	TRIGGER
RENAME	SEQUENCE
COMMENT ON[2]	MATERIALIZED VIEW
	VIEW
	FUNCTION
	PACKAGE
	PROCEDURE
	SYNONYM
	PUBLIC SYNONYM[4]
GRANT	TABLE
REVOKE	SEQUENCE
	MATERIALIZED VIEW

Table 13-1 (Cont.) Objects That Can be Mapped in MAP and TABLE Statements

Operations	On any of these Objects[1]
ANALYZE	TABLE
	INDEX
	CLUSTER

1 TABLE and MAP do not support some special characters that could be used in an object name affected by these operations. Objects with non-supported special characters are supported by the scopes of UNMAPPED and OTHER.
2 Applies to COMMENT ON TABLE, COMMENT ON COLUMN
3 Includes AS SELECT
4 Table name must be qualified with schema name.

For Extract, MAPPED scope marks an object for DDL capture according to the instructions in the TABLE statement. For Replicat, MAPPED scope marks DDL for replication and maps it to the object specified by the schema and name in the TARGET clause of the MAP statement. To perform this mapping, Replicat issues ALTER SESSION to set the schema of the Replicat session to the schema that is specified in the TARGET clause. If the DDL contains unqualified objects, the schema that is assigned on the target depends on circumstances described in Correctly Identifying Unqualified Object Names in DDL (page 13-11).

Assume the following TABLE and MAP statements:

Extract (source)

```
TABLE fin.expen;
TABLE hr.tab*;
```

Replicat (target)

```
MAP fin.expen, TARGET fin2.expen2;
MAP hr.tab*, TARGET hrBackup.bak_*;
```

Also assume a source DDL statement of:

```
ALTER TABLE fin.expen ADD notes varchar2(100);
```

In this example, because the source table fin.expen is in a MAP statement with a TARGET clause that maps to a different schema and table name, the target DDL statement becomes:

```
ALTER TABLE fin2.expen2 ADD notes varchar2(100);
```

Likewise, the following source and target DDL statements are possible for the second set of TABLE and MAP statements in the example:

Source:

```
CREATE TABLE hr.tabPayables ... ;
```

Target:

```
CREATE TABLE hrBackup.bak_tabPayables ...;
```

When objects are of MAPPED scope, you can omit their names from the DDL configuration parameters, unless you want to refine their DDL support further. If you ever need to change the object names in TABLE and MAP statements, the changes will apply automatically to the DDL on those objects.

If you include an object in a TABLE statement, but not in a MAP statement, the DDL for that object is MAPPED in scope on the source but UNMAPPED in scope on the target.

- Mapping Oracle Cluster Tables and UDTs (page 13-10)
- Mapping ALTER INDEX (page 13-10)

13.5.1.1 Mapping Oracle Cluster Tables and UDTs

An Oracle clustered table or Oracle user defined type (UDT) cannot be mapped to a different target name, but it can be mapped to a different target schema. Because these special kinds of objects can consist of underlying tables that, themselves, could be a mix of both MAPPED and UNMAPPED scope, name mapping cannot be used.

13.5.1.2 Mapping ALTER INDEX

An ALTER INDEX...RENAME command cannot be mapped to a different target index name, but it can be mapped to a different target schema.

Valid example:

```
ALTER INDEX src.ind RENAME TO indnew;
```

This DDL can be mapped with wildcards as:

```
MAP src.* TARGET tgt.*;
```

Alternatively, it can be mapped explicitly as the following, making sure to use the original index name in the source and target specifications:

```
MAP src.ind TARGET tgt.ind;
```

In either of the preceding cases, the target DDL will be:

```
ALTER INDEX tgt.ind RENAME TO indnew;
```

Invalid example:

A MAP statement such as the following is not valid:

```
MAP src.ind TARGET tgt.indnew;
```

That statement maps the old name to the new name, and the target DDL will become:

```
ALTER INDEX tgt.indnew RENAME TO indnew;
```

13.5.2 Unmapped Scope

If a DDL operation is supported for use in a TABLE or MAP statement, but its base object name is not included in one of those parameters, it is of UNMAPPED scope.

An object name can be of UNMAPPED scope on the source (not in an Extract TABLE statement), but of MAPPED scope on the target (in a Replicat MAP statement), or the other

way around. When Oracle DDL is of `UNMAPPED` scope in the Replicat configuration, Replicat will by default do the following:

1. Set the current schema of the Replicat session to the schema of the source DDL object.
2. Execute the DDL as that schema.
3. Restore Replicat as the current schema of the Replicat session.

See Correctly Identifying Unqualified Object Names in DDL (page 13-11).

13.5.3 Other Scope

DDL operations that cannot be mapped are of `OTHER` scope. When DDL is of `OTHER` scope in the Replicat configuration, it is applied to the target with the same schema and object name as in the source DDL.

An example of `OTHER` scope is a DDL operation that makes a system-specific reference, such as DDL that operates on data file names.

Some other examples of `OTHER` scope:

```
CREATE USER joe IDENTIFIED by joe;
CREATE ROLE ggs_gguser_role IDENTIFIED GLOBALLY;
ALTER TABLESPACE gg_user TABLESPACE GROUP gg_grp_user;
```

See Correctly Identifying Unqualified Object Names in DDL (page 13-11).

13.6 Correctly Identifying Unqualified Object Names in DDL

Extract captures the current schema (also called session schema) that is in effect when a DDL operation is executed. The current container is also captured if the source is a multitenant container database. The container and schema are used to resolve unqualified object names in the DDL.

Consider the following example:

```
CONNECT SCOTT/TIGER
CREATE TABLE TAB1 (X NUMBER);
CREATE TABLE SRC1.TAB2(X NUMBER) AS SELECT * FROM TAB1;
```

In both of those DDL statements, the unqualified table `TAB1` is resolved as `SCOTT.TAB1` based on the current schema `SCOTT` that is in effect during the DDL execution.

There is another way of setting the current schema, which is to set the `current_schema` for the session, as in the following example:

```
CONNECT SCOTT/TIGER
ALTER SESSION SET CURRENT_SCHEMA=SRC;
CREATE TABLE TAB1 (X NUMBER);
CREATE TABLE SRC1.TAB2(X NUMBER) AS SELECT * FROM TAB1;
```

In both of those DDL statements, the unqualified table `TAB1` is resolved as `SRC.TAB1` based on the current schema `SRC` that is in effect during the DDL execution.

In both classic and integrated capture modes, Extract captures the current schema that is in effect during DDL execution, and it resolves the unqualified object names (if any) by using the current schema. As a result, `MAP` statements specified for Replicat work correctly for DDL with unqualified object names.

You can also map a source session schema to a different target session schema, if that is required for the DDL to succeed on the target. This mapping is global and overrides any other mappings that involve the same schema names. To map session schemas, use the `DDLOPTIONS` parameter with the `MAPSESSIONSCHEMA` option. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.

If the default or mapped session schema mapping fails, you can handle the error with the following `DDLERROR` parameter statement, where error 1435 means that the schema does not exist.

```
DDLERROR 1435 IGNORE INCLUDE OPTYPE ALTER OBJTYPE SESSION
```

13.7 Enabling DDL Support

By default, the status of DDL replication support is as follows:

- On the source, Oracle GoldenGate DDL support is disabled by default. You must configure Extract to capture DDL by using the `DDL` parameter.
- On the target, DDL support is enabled by default, to maintain the integrity of transactional data that is replicated. By default, Replicat will process all DDL operations that the trail contains. If needed, you can use the `DDL` parameter to configure Replicat to ignore or filter DDL operations.

13.8 Filtering DDL Replication

You can use the following methods to filter DDL operations so that specific (or all) DDL is applied to the target database according to your requirements. By default, all DDL is passed to Extract.

- Filtering with PL/SQL Code (page 13-13): Valid only for trigger-based DDL capture. This method makes use of an Oracle function that is called by the DDL trigger when a DDL operation occurs, to compute whether or not to send the DDL to Extract. Filtering with PL/SQL code should only be used to improve the performance of the source database when the DDL trigger is in use. It can be combined with built-in rules and DDL parameter filtering (see the following). Any DDL that is passed to Extract after it is filtered by the DDL trigger or filter rules can be filtered further with the `DDL` parameter to meet specific needs.
- Filtering With Built-in Filter Rules (page 13-15): Valid only for trigger-based DDL capture. This method makes use of some procedures that you run to build filter rules into the Oracle GoldenGate trigger logic. This method allows discreet control over the types of objects that are sent to Extract, and it allows the ordering of rule evaluation. This method should only be used to improve the performance of the source database when the DDL trigger is in use. You can combine built-in rules with PL/SQL and DDL parameter filtering. Any DDL that is passed to Extract after it is filtered by the DDL trigger or filter rules can be filtered further with the `DDL` parameter to meet specific needs.

> **Note:**
>
> Filtering with PL/SQL or built-in filter rules is unnecessary for an Extract that operates in integrated-capture mode. If Extract must operate in classic mode and you use these filtering methods, the same filtering must happen for any transactional data (DML) that is associated with the filtered objects. For example, if you filter out the DDL that creates a table named `ACCOUNTS`, make certain the `ACCOUNTS` table is not specified in any `TABLE` or `MAP` statements, or use the appropriate exclusion parameter to exclude it from wildcard resolution. See *Reference for Oracle GoldenGate for Windows and UNIX* for a list of wildcard exclusion parameters.

- Filtering with the DDL Parameter (page 13-18):Valid for both trigger-based and native DDL capture. This is the preferred method of filtering and is performed within Oracle GoldenGate, and both Extract and Replicat can execute filter criteria. Extract can perform filtering, or it can send all of the DDL to a trail, and then Replicat can perform the filtering. Alternatively, you can filter in a combination of different locations. The `DDL` parameter gives you control over where the filtering is performed, and it also offers more filtering options than the trigger method, including the ability to filter collectively based on the DDL scope (for example, include all `MAPPED` scope).

> **Note:**
>
> If a DDL operation fails in the middle of a `TRANSACTION`, it forces a commit, which means that the transaction spanning the DDL is split into two. The first half is committed and the second half can be restarted. If a recovery occurs, the second half of the transaction cannot be filtered since the information contained in the header of the transaction is no longer there.

- Filtering with PL/SQL Code (page 13-13)
- Filtering With Built-in Filter Rules (page 13-15)
- Filtering with the DDL Parameter (page 13-18)

13.8.1 Filtering with PL/SQL Code

This method is only valid for trigger-based capture.

You can write PL/SQL code to pass information about the DDL to a function that computes whether or not the DDL is passed to Extract. By sending fewer DDL operations to Extract, you can improve capture performance.

1. Copy the `ddl_filter.sql` file that is in the Oracle GoldenGate installation directory to a test machine where you can test the code that you will be writing.
2. Open the file for editing. It contains a PL/SQL function named `filterDDL`, which you can modify to specify `if/then` filter criteria. The information that is passed to this function includes:
   - `ora_owner`: the schema of the DDL object

- `ora_name`: the defined name of the object
- `ora_objtype`: the type of object, such as `TABLE` or `INDEX`
- `ora_optype`: the operation type, such as `CREATE` or `ALTER`
- `ora_login_user`: The user that executed the DDL
- `retVal`: can be either `INCLUDE` to include the DDL, or `EXCLUDE` to exclude the DDL from Extract processing.

In the location after the `'compute retVal here'` comment, write filter code for each type of DDL that you want to be filtered. The following is an example:

```
if ora_owner='SYS' then
retVal:='EXCLUDE';
end if;
if ora_objtype='USER' and ora_optype ='DROP' then
retVal:='EXCLUDE';
end if;
if ora_owner='JOE' and ora_name like 'TEMP%' then
retVal:='EXCLUDE';
end if;
```

In this example, the following DDL is excluded from being processed by the DDL trigger:

- DDL for objects owned by `SYS`
- any `DROP USER`
- any DDL on `JOE.TEMP%`

3. (Optional) To trace the filtering, you can add the following syntax to each `if/then` statement in the PL/SQL:

```
if ora_owner='JOE' and ora_name like 'TEMP%' then
retVal:='EXCLUDE';
if "&gg_user" .DDLReplication.trace_level >= 1 then
"&gg_user" .trace_put_line ('DDLFILTER', 'excluded JOE.TEMP%');
end if;
```

Where:

- `&gg_user` is the schema of the Oracle GoldenGate DDL support objects.
- `.DDLReplication.trace_level` is the level of DDL tracing. To use trigger tracing, the `TRACE` or `TRACE2` parameter must be used with the `DDL` or `DDLONLY` option in the Extract parameter file. The `.DDLReplication.trace_level` parameter must be set to `>=1`.
- `trace_put_line` is a user-defined text string that Extract writes to the trace file that represents the type of DDL that was filtered.

4. Save the code.
5. Stop DDL activity on the test system.
6. In SQL*Plus, compile the `ddl_filter.sql` file as follows, where `schema_name` is the schema where the Oracle GoldenGate DDL objects are installed.

```
@ddl_filter schema_name
```

7. Test in the test environment to make certain that the filtering works. It is important to perform this testing, because any errors in the code could cause source and target DDL to become out of synchronization.
8. After a successful test, copy the file to the Oracle GoldenGate installation directory on the source production system.
9. Stop DDL activity on the source system.
10. Compile the `ddl_filter.sql` file as you did before.

    ```
    @ddl_filter schema_name
    ```
11. Resume DDL activity on the source system.

13.8.2 Filtering With Built-in Filter Rules

This method is only valid for trigger-based capture.

You can add inclusion and exclusion rules to control the DDL operations that are sent to Extract by the DDL trigger. By storing rules and sending fewer DDL operations to Extract, you can improve capture performance.

1. Use the `DDLAUX.addRule()` function to define your rules according to the following instructions. This function is installed in the Oracle GoldenGate DDL schema after the DDL objects are installed with the `ddl_setup.sql` script.
2. To activate the rules, execute the function in SQL*Plus or enter a collection of rules in a SQL file and execute that file in SQL*Plus.

- DDLAUX.addRule() Function Definition (page 13-15)
- Parameters for DDLAUX.addRule() (page 13-15)
- Valid DDL Components for DDLAUX.addRule() (page 13-16)
- Examples of Rule-based Trigger Filtering (page 13-17)
- Dropping Filter Rules (page 13-18)

13.8.2.1 `DDLAUX.addRule()` Function Definition

```
FUNCTION addRule( obj_name IN VARCHAR2 DEFAULT NULL,
base_obj_name IN VARCHAR2 DEFAULT NULL,
owner_name IN VARCHAR2 DEFAULT NULL,
base_owner_name IN VARCHAR2 DEFAULT NULL,
base_obj_property IN NUMBER DEFAULT NULL,
obj_type IN NUMBER DEFAULT NULL,
command IN VARCHAR2 DEFAULT NULL,
inclusion IN boolean DEFAULT NULL ,
sno IN NUMBER DEFAULT NULL)
RETURN NUMBER;
```

13.8.2.2 Parameters for `DDLAUX.addRule()`

The information passed to this function are the following parameters, which correlate to the attributes of an object. All parameters are optional, and more than one parameter can be specified.

- `sno`: Specifies a serial number that identifies the rule. The order of evaluation of rules is from the lowest serial number to the highest serial number, until a match is

found. The `sno` can be used to place inclusion rules ahead of an exclusion rule, so as to make an exception to the exclusion rule. Because this is a function and not a procedure, it returns the serial number of the rule, which should be used for the drop rule specified with `DDLAUX.dropRule()`. The serial number is generated automatically unless you specify one with this statement at the beginning of your code: `DECLARE sno NUMBER; BEGIN sno :=`

For example:

```
DECLARE
  sno NUMBER;
BEGIN
  sno := tkggadmin..DDLAUX.ADDRULE(obj_name => 'GGS%' ,
                                   obj_type => TYPE_TABLE);
END;
/
```

- `obj_name`: Specifies the object name. If the name is case-sensitive, enclose it within double quotes.
- `owner_name`: Specifies the name of the object schema
- `base_obj_name`: Specifies the base object name of the DDL object (such as the base table if the object is an index). If the name is case-sensitive, enclose it within double quotes.
- `base_owner_name`: Specifies the base object schema name.
- `base_obj_property`: Specifies the base object property. See Valid DDL Components for DDLAUX.addRule() (page 13-16)
- `obj_type`: Specifies the object type. See Valid DDL Components for DDLAUX.addRule() (page 13-16)
- `command`: Specifies the command. See Valid DDL Components for DDLAUX.addRule() (page 13-16)
- `inclusion = TRUE`: Indicates that the specified objects are to be captured by the DDL trigger. If this parameter is not specified, the rule becomes an exclusion rule, and the specified objects are not captured. You can specify both an exclusion rule and an inclusion rule. If a DDL does not match any of the rules, it is included (passed to Extract) by default. Calling `DDLAUX.addRule()` without any parameters generates an *empty rule* that excludes all DDL on all the objects.

13.8.2.3 Valid DDL Components for DDLAUX.addRule()

The following are the defined DDL object types, base object properties, and DDL commands that can be specified in the function code.

Valid object types are:

```
TYPE_INDEX
TYPE_TABLE
TYPE_VIEW
TYPE_SYNONYM
TYPE_SEQUENCE
TYPE_PROCEDURE
TYPE_FUNCTION
TYPE_PACKAGE
TYPE_TRIGGER
```

Valid base object properties are:

```
TB_IOT
TB_CLUSTER
TB_NESTED
TB_TEMP
TB_EXTERNAL
```

Valid commands are:

```
CMD_CREATE
CMD_DROP
CMD_TRUNCATE
CMD_ALTER
```

13.8.2.4 Examples of Rule-based Trigger Filtering

The following example excludes all temporary tables, except tables with names that start with `IMPTEMP`.

```
1. DDLAUX.ADDRULE(obj_name => 'IMPTEMP%', base_obj_property => TB_TEMP, obj_type =>
TYPE_TABLE, INCLUSION => TRUE);
2. DDLAUX.ADDRULE(base_obj_property => TB_TEMP, obj_type => TYPE_TABLE);
```

> **Note:**
>
> Since the `IMPTEMP%` tables must be included, that rule should come first.

The following example excludes all tables with name 'GGS%'

```
DECLARE sno NUMBER; BEGIN sno := DDLAUX.ADDRULE(obj_name => 'GGS%' , obj_type =>
TYPE_TABLE); END
```

The following example excludes all temporary tables.

```
DDLAUX.ADDRULE(base_obj_property => TB_TEMP, obj_type => TYPE_TABLE);
```

The following example excludes all indexes on `TEMP` tables.

```
DDLAUX.ADDRULE(base_obj_property => TB_TEMP, obj_type => TYPE_INDEX);
```

The following example excludes all objects in schema `TKGGADMIN`.

```
DDLAUX.ADDRULE(owner_name => 'TKGGADMIN');
```

The following example excludes all objects in `TRUNCATE` operations made to `TEMP` tables.

```
DDLAUX.ADDRULE(base_obj_property => TB_TEMP, obj_type => TYPE_TABLE, command =>
CMD_TRUNCATE)
```

13.8.2.5 Dropping Filter Rules

Use the `DDLAUX.dropRule()` function with the drop rule. This function is installed in the Oracle GoldenGate DDL schema after the DDL objects are installed with the `ddl_setup.sql` script. As input, specify the serial number of the rule that you want to drop.

```
FUNCTION dropRule(sno IN NUMBER) RETURN BOOLEAN;
```

13.8.3 Filtering with the DDL Parameter

This method is valid for trigger-based and integrated capture modes.

The `DDL` parameter is the main Oracle GoldenGate parameter for filtering DDL within the Extract and Replicat processes.

When used without options, the `DDL` parameter performs no filtering, and it causes all DDL operations to be propagated as follows:

- As an Extract parameter, it captures all supported DDL operations that are generated on all supported database objects and sends them to the trail.
- As a Replicat parameter, it replicates all DDL operations from the Oracle GoldenGate trail and applies them to the target. This is the same as the default behavior without this parameter.

When used with options, the `DDL` parameter acts as a filtering agent to include or exclude DDL operations based on:

- scope
- object type
- operation type
- object name
- strings in the DDL command syntax or comments, or both

Only one `DDL` parameter can be used in a parameter file, but you can combine multiple inclusion and exclusion options, along with other options, to filter the DDL to the required level.

- `DDL` filtering options are valid for a primary Extract that captures from the transaction source, but not for a data-pump Extract.
- When combined, multiple filter option specifications are linked logically as `AND` statements.
- All filter criteria specified with multiple options must be satisfied for a DDL statement to be replicated.
- When using complex DDL filtering criteria, it is recommended that you test your configuration in a test environment before using it in production.

For `DDL` parameter syntax and additional usage guidelines, see *Reference for Oracle GoldenGate for Windows and UNIX*.

> **Note:**
>
> Before you configure DDL support, it might help to review How DDL is Evaluated for Processing (page 13-29).

13.9 Special Filter Cases

The following are special cases that you should be aware of when creating your filter conditions.

13.9.1 DDL EXCLUDE ALL

`DDL EXCLUDE ALL` is a special processing option that is intended primarily for Extract when using trigger-based DDL capture. `DDL EXCLUDE ALL` blocks the replication of DDL operations, but ensures that Oracle GoldenGate continues to keep the object metadata current. When Extract receives DDL directly from the logmining server (triggerless DDL capture mode), current metadata is always maintained.

You can use `DDL EXCLUDE ALL` when using a method other than Oracle GoldenGate to apply DDL to the target and you want Oracle GoldenGate to replicate data changes to the target objects. It provides the current metadata to Oracle GoldenGate as objects change, thus preventing the need to stop and start the Oracle GoldenGate processes. The following special conditions apply to `DDL EXCLUDE ALL`:

- `DDL EXCLUDE ALL` does not require the use of an `INCLUDE` clause.
- When using `DDL EXCLUDE ALL`, you can set the `WILDCARDRESOLVE` parameter to `IMMEDIATE` to allow immediate DML resolution if required. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.

To prevent all DDL metadata and operations from being replicated, omit the `DDL` parameter entirely.

13.9.2 Implicit DDL

User-generated DDL operations can generate implicit DDL operations. For example, the following statement generates two distinct DDL operations.

```
CREATE TABLE customers (custID number, name varchar2(50), address varchar2(75),
address2 varchar2(75), city varchar2(50), state (varchar2(2), zip number, contact
varchar2(50), areacode number(3), phone number(7), primary key (custID));
```

The first (explicit) DDL operation is the `CREATE TABLE` statement itself.

The second DDL operation is an implicit `CREATE UNIQUE INDEX` statement that creates the index for the primary key. This operation is generated by the database engine, not a user application.

Guidelines for Filtering Implicit DDL

How to filter implicit DDL depends on the mechanism that you are using to filter DDL. See Filtering DDL Replication (page 13-12) for more information.

- When the `DDL` parameter is used to filter DDL operations, Oracle GoldenGate filters out any implicit DDL by default, because the explicit DDL will generate the implicit DDL on the target. For example, the target database will create the appropriate index when the `CREATE TABLE` statement in the preceding example is applied by Replicat.
- When the DDL trigger is being used to filter DDL operations, you must handle the implicit DDL in your filter rules based on the following:
  - If your filtering rules exclude the explicit DDL from being propagated, you must also create a rule to exclude the implicit DDL. For example, if you exclude the `CREATE TABLE` statement in the following example, but do not exclude the implicit `CREATE UNIQUE INDEX` statement, the target database will try to create the index on a non-existent table.

    ```
    CREATE TABLE customers (custID number, name varchar2(50), address
    varchar2(75), address2 varchar2(75), city varchar2(50), state (varchar2(2),
    zip number, contact varchar2(50), areacode number(3), phone number(7),
    primary key (custID));
    ```

  - If your filtering rules permit the propagation of the explicit DDL, you do not need to exclude the implicit DDL. It will be handled correctly by Oracle GoldenGate and the target database.

13.10 How Oracle GoldenGate Handles Derived Object Names

DDL operations can contain a *base object* name and also a *derived object* name. A base object is an object that contains data. A derived object is an object that inherits some attributes of the base object to perform a function related to that object. DDL statements that have both base and derived objects are:

- `RENAME` and `ALTER RENAME`
- `CREATE` and `DROP` on an index, synonym, or trigger

Consider the following DDL statement:

```
CREATE INDEX hr.indexPayrollDate ON TABLE hr.tabPayroll (payDate);
```

In this case, the table is the base object. Its name (`hr.tabPayroll`) is the *base name* and is subject to mapping with `TABLE` or `MAP` under the `MAPPED` scope. The derived object is the index, and its name (`hr.indexPayrollDate`) is the *derived name*.

You can map a derived name in its own `TABLE` or `MAP` statement, separately from that of the base object. Or, you can use one `MAP` statement to handle both. In the case of `MAP`, the conversion of derived object names on the target works as follows.

- MAP Exists for Base Object, But Not Derived Object (page 13-21)
- MAP Exists for Base and Derived Objects (page 13-21)
- MAP Exists for Derived Object, But Not Base Object (page 13-22)

- New Tables as Derived Objects (page 13-22)
- Disabling the Mapping of Derived Objects (page 13-24)

13.10.1 MAP Exists for Base Object, But Not Derived Object

If there is a `MAP` statement for the base object, but not for the derived object, the result is a schema based on the mapping that matches the derived object name. Derived objects are only mapped if the `MAPDERIVED` option is enabled, which is also the default option.

For example, consider the following:

Extract (source)

```
Table hr.*;
```

Replicat (target)

```
MAP hr.*, TARGET hrBackup.*;
```

Assume the following source DDL statement:

```
CREATE INDEX hr.indexPayrollDate ON TABLE hr.Payroll (payDate);
```

The `CREATE INDEX` statement is executed by Replicat on the target as follows:

```
CREATE INDEX hrBackup.indexPayrollDate ON TABLE hrBackup.Payroll (payDate);
```

In this example, the mapping is such that it matches the derived object name because of which the derived object schema is changed from `hr` to `hrBackup`.

Here's another example, where there is no mapping that matches the derived object name so the derived object name remains the same.

Extract (source)

```
Table hr.tab*;
```

Replicat (target)

```
MAP hr.tab*, TARGET hrBackup.*;
```

Assume the following source DDL statement:

```
CREATE INDEX hr.indexPayrollDate ON TABLE hr.tabPayroll (payDate);
```

The `CREATE INDEX` statement is executed by Replicat on the target as follows:

```
CREATE INDEX hr.indexPayrollDate ON TABLE hrBackup.tabPayroll (payDate);
```

13.10.2 MAP Exists for Base and Derived Objects

If there is a `MAP` statement for the base object and also one for the derived object, the result is an explicit mapping. Assuming the DDL statement includes `MAPPED`, Replicat converts the schema and name of each object according to its own `TARGET` clause. For example, assume the following:

Extract (source)

```
TABLE hr.tab*;  TABLE hr.index*;
```

Replicat (target)

```
MAP hr.tab*, TARGET hrBackup.*;MAP hr.index*, TARGET hrIndex.*;
```

Assume the following source DDL statement:

```
CREATE INDEX hr.indexPayrollDate ON TABLE hr.tabPayroll (payDate);
```

The `CREATE INDEX` statement is executed by Replicat on the target as follows:

```
CREATE INDEX hrIndex.indexPayrollDate ON TABLE hrBackup.tabPayroll (payDate);
```

Use an explicit mapping when the index on the target must be owned by a different schema from that of the base object, or when the name on the target must be different from that of the source.

13.10.3 MAP Exists for Derived Object, But Not Base Object

If there is a `MAP` statement for the derived object, but not for the base object, Replicat does not perform any name conversion for either object. The target DDL statement is the same as that of the source. To map a derived object, the choices are:

- Use an explicit `MAP` statement for the base object.
- If names permit, map both base and derived objects in the same `MAP` statement by means of a wildcard.
- Create a `MAP` statement for each object, depending on how you want the names converted.

13.10.4 New Tables as Derived Objects

The following explains how Oracle GoldenGate handles new tables that are created from:

- `RENAME` and `ALTER RENAME`
- `CREATE TABLE AS SELECT`
- RENAME and ALTER TABLE RENAME (page 13-22)
- CREATE TABLE AS SELECT (page 13-23)

13.10.4.1 RENAME and ALTER TABLE RENAME

In `RENAME` and `ALTER TABLE RENAME` operations, the base object is always the new table name. In the following example, the base object name is considered to be `index_paydate`.

```
ALTER TABLE hr.indexPayrollDate RENAME TO index_paydate;
```

or...

```
RENAME hr.indexPayrollDate TO index_paydate;
```

The derived object name is `hr.indexPayrollDate`.

13.10.4.2 CREATE TABLE AS SELECT

The `CREATE TABLE AS SELECT` (CTAS) statements include `SELECT` statements and `INSERT` statements that reference any number of underlying objects. By default, Oracle GoldenGate obtains the data for the `AS SELECT` clause from the target database. You can force the CTAS operation to preserve the original inserts using this parameter.

> **Note:**
>
> For this reason, Oracle `XMLType` tables created from a `CTAS` (`CREATE TABLE AS SELECT`) statement cannot be supported. For `XMLType` tables, the row object IDs must match between source and target, which cannot be maintained in this scenario. `XMLType` tables created by an empty `CTAS` statement (that does not insert data in the new table) can be maintained correctly.
>
> In addition, you could use the `GETCTASDML` parameter that allows CTAS to replay the inserts of the CTAS thus preserving OIDs during replication. This parameter is only supported with Integrated Dictionary and any downstream Replicat must be 12.1.2.1 or greater to consume the trail otherwise, there may be divergence.

The objects in the `AS SELECT` clause must exist in the target database, and their names must be identical to the ones on the source.

In a `MAP` statement, Oracle GoldenGate only maps the name of the new table (`CREATE TABLE name`) to the `TARGET` specification, but does not map the names of the underlying objects from the `AS SELECT` clause. There could be dependencies on those objects that could cause data inconsistencies if the names were converted to the `TARGET` specification.

The following shows an example of a `CREATE TABLE AS SELECT` statement on the source and how it would be replicated to the target by Oracle GoldenGate.

```
CREATE TABLE a.tab1 AS SELECT * FROM a.tab2;
```

The `MAP` statement for Replicat is as follows:

```
MAP a.tab*, TARGET a.x*;
```

The target DDL statement that is applied by Replicat is the following:

```
CREATE TABLE a.xtab1 AS SELECT * FROM a.tab2;
```

The name of the table in the `AS SELECT * FROM` clause remains as it was on the source: `tab2` (rather than `xtab2`).

To keep the data in the underlying objects consistent on source and target, you can configure them for data replication by Oracle GoldenGate. In the preceding example, you could use the following statements to accommodate this requirement:

Source

```
TABLE a.tab*;
```

Target

```
MAPEXCLUDE a.tab2
MAP a.tab*, TARGET a.x*;
MAP a.tab2, TARGET a.tab2;
```

See Correctly Identifying Unqualified Object Names in DDL (page 13-11).

13.10.5 Disabling the Mapping of Derived Objects

Use the DDLOPTIONS parameter with the NOMAPDERIVED option to prevent the conversion of the name of a derived object according to a TARGET clause of a MAP statement that includes it. NOMAPDERIVED overrides any explicit MAP statements that contain the name of the base or derived object. Source DDL that contains derived objects is replicated to the target with the same schema and object names as on the source.

The following table shows the results of MAPDERIVED compared to NOMAPDERIVED, based on whether there is a MAP statement just for the base object, just for the derived object, or for both.

Table 13-2 [NO]MAPDERIVED Results on Target Based on Mapping Configuration

Base Object	Derived Object	MAP/NOMAP DERIVED?	Derived object converted per a MAP?	Derived object gets schema of base object?
mapped[1]	mapped	MAPDERIVED	yes	no
mapped	not mapped	MAPDERIVED	no	yes
not mapped	mapped	MAPDERIVED	no	no
not mapped	not mapped	MAPDERIVED	no	no
mapped	mapped	NOMAPDERIVED	no	no
mapped	not mapped	NOMAPDERIVED	no	no
not mapped	mapped	NOMAPDERIVED	no	no
not mapped	not mapped	NOMAPDERIVED	no	no

1 Mapped means included in a MAP statement.

The following examples illustrate the results of MAPDERIVED as compared to NOMAPDERIVED.In Table 13-3 (page 13-24), both trigger and table are owned by rpt on the target because both base and derived names are converted by means of MAPDERIVED.

Table 13-3 Default Mapping of Derived Object Names (MAPDERIVED)

MAP statement	Source DDL statement captured by Extract	Target DDL statement applied by Replicat
MAP fin.*, TARGET rpt.*;	CREATE TRIGGER fin.act_trig ON fin.acct;	CREATE TRIGGER rpt.act_trig ON rpt.acct;

In Table 13-4 (page 13-25), the trigger is owned by fin, because conversion is prevented by means of NOMAPDERIVED.

Table 13-4 Mapping of derived object names when using NOMAPDERIVED

MAP statement	Source DDL statement captured by Extract	Target DDL statement applied by Replicat
`MAP fin.*, TARGET rpt.*;`	`CREATE TRIGGER fin.act_trig ON fin.acct;`	`CREATE TRIGGER fin.act_trig ON rpt.acct;`

> **Note:**
>
> In the case of a `RENAME` statement, the new table name is considered to be the base table name, and the old table name is considered to be the derived table name.

13.11 Using DDL String Substitution

You can substitute strings within a DDL operation while it is being processed by Oracle GoldenGate. This feature provides a convenience for changing and mapping directory names, comments, and other things that are not directly related to data structures. For example, you could substitute one tablespace name for another, or substitute a string within comments. String substitution is controlled by the `DDLSUBST` parameter. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.

> **Note:**
>
> Before you create a `DDLSUBST` parameter statement, it might help to review How DDL is Evaluated for Processing (page 13-29) in this chapter.

13.12 Controlling the Propagation of DDL to Support Different Topologies

To support bidirectional and cascading replication configurations, it is important for Extract to be able to identify the DDL that is performed by Oracle GoldenGate and by other applications, such as the local business applications. Depending on the configuration that you want to deploy, it might be appropriate to capture one or both of these sources of DDL on the local system.

> **Note:**
>
> Oracle GoldenGate DDL consists of `ALTER TABLE` statements performed by Extract to create log groups and the DDL that is performed by Replicat to replicate source DDL changes.

The following options of the DDLOPTIONS parameter control whether DDL on the local system is captured by Extract and then sent to a remote system, assuming Oracle GoldenGate DDL support is configured and enabled:

- The GETREPLICATES and IGNOREREPLICATES options control whether Extract captures or ignores the DDL that is generated by Oracle GoldenGate. The default is IGNOREREPLICATES, which does not propagate the DDL that is generated by Oracle GoldenGate. To identify the DDL operations that are performed by Oracle GoldenGate, the following comment is part of each Extract and Replicat DDL statement:

  ```
  /* GOLDENGATE_DDL_REPLICATION */
  ```

- The GETAPPLOPS and IGNOREAPPLOPS options control whether Extract captures or ignores the DDL that is generated by applications other than Oracle GoldenGate. The default is GETAPPLOPS, which propagates the DDL from local applications (other than Oracle GoldenGate).

The result of these default settings is that Extract ignores its own DDL and the DDL that is applied to the local database by a local Replicat, so that the DDL is not sent back to its source, and Extract captures all other DDL that is configured for replication. The following is the default DDLOPTIONS configuration.

```
DDLOPTIONS GETAPPLOPS, IGNOREREPLICATES
```

This behavior can be modified. See the following topics:

- Propagating DDL in Active-Active (Bidirectional) Configurations (page 13-26)
- Propagating DDL in a Cascading Configuration (page 13-28)

- Propagating DDL in Active-Active (Bidirectional) Configurations (page 13-26)
- Propagating DDL in a Cascading Configuration (page 13-28)

13.12.1 Propagating DDL in Active-Active (Bidirectional) Configurations

Oracle GoldenGate supports active-active DDL replication between two systems. For an active-active bidirectional replication, the following must be configured in the Oracle GoldenGate processes:

1. DDL that is performed by a business application on one system must be replicated to the other system to maintain synchronization. To satisfy this requirement, include the GETAPPLOPS option in the DDLOPTIONS statement in the Extract parameter files on both systems.
2. DDL that is applied by Replicat on one system must be captured by the local Extract and sent back to the other system. To satisfy this requirement, use the GETREPLICATES option in the DDLOPTIONS statement in the Extract parameter files on both systems.

> **Note:**
>
> An internal Oracle GoldenGate token will cause the actual Replicat DDL statement itself to be ignored to prevent loopback. The purpose of propagating Replicat DDL back to the original system is so that the Replicat on that system can update its object metadata cache, in preparation to receive incoming DML, which will have the new metadata. See Figure 13-1 (page 13-27).

3. Each Replicat must be configured to update its object metadata cache whenever the remote Extract sends over a captured Replicat DDL statement. To satisfy this requirement, use the `UPDATEMETADATA` option in the `DDLOPTIONS` statement in the Replicat parameter files on both systems.

The resultant `DDLOPTIONS` statements should look as follows:

Extract (primary and secondary)

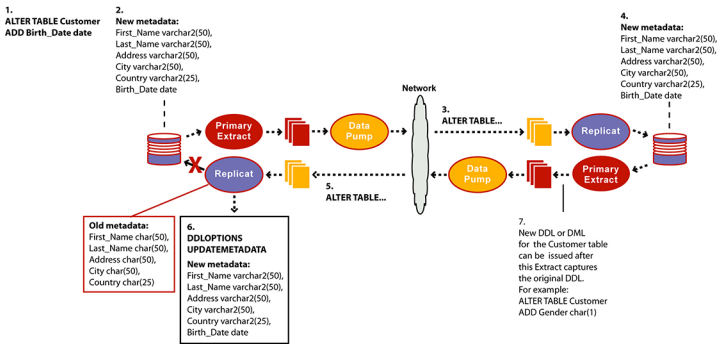
```
DDLOPTIONS GETREPLICATES, GETAPPLOPS
```

Replicat (primary and secondary)

```
DDLOPTIONS UPDATEMETADATA
```

> **WARNING:**
>
> Before you allow new DDL or DML to be issued for the same object(s) as the original DDL, allow time for the original DDL to be replicated to the remote system and then captured again by the Extract on that system. This will ensure that the operations arrive in correct order to the Replicat on the original system, to prevent DML errors caused by metadata inconsistencies. See Figure 13-1 (page 13-27) for more information.

Figure 13-1 Path of DDL in Round Trip to Update Replicat Object Metadata Cache



For more information about `DDLOPTIONS`, see *Reference for Oracle GoldenGate for Windows and UNIX*.

For more information about configuring a bidirectional configuration, see *Administering Oracle GoldenGate for Windows and UNIX*.

13.12.2 Propagating DDL in a Cascading Configuration

In a cascading configuration, use the following setting for DDLOPTIONS in the Extract parameter file on each intermediary system. This configuration forces Extract to capture the DDL from Replicat on an intermediary system and cascade it to the next system downstream.

```
DDLOPTIONS GETREPLICATES, IGNOREAPPLOPS
```

For more information about DDLOPTIONS, see *Reference for Oracle GoldenGate for Windows and UNIX*

For more information about configuring a cascading configuration, see Administering Oracle GoldenGate for Windows and UNIX.

13.13 Adding Supplemental Log Groups Automatically

Use the DDLOPTIONS parameter with the ADDTRANDATA option to do the following:

- enable Oracle's supplemental logging automatically for new tables created with a CREATE TABLE.
- update Oracle's supplemental logging for tables affected by an ALTER TABLE to add or drop columns.
- update Oracle's supplemental logging for tables that are renamed.
- update Oracle's supplemental logging for tables where unique or primary keys are added or dropped.

To use DDLOPTIONS ADDSCHEMATRANDATA, the ADD SCHEMATRANDATA command must be issued in GGSCI to enable schema-level supplemental logging.

By default, the ALTER TABLE that adds the supplemental logging is not replicated to the target unless the GETREPLICATES parameter is in use.

DDLOPTIONS ADDTRANDATA is not supported for multitenant container databases.

See Configuring Logging Properties (page 3-2) for more information.

13.14 Removing Comments from Replicated DDL

You can use the DDLOPTIONS parameter with the REMOVECOMMENTS BEFORE and REMOVECOMMENTS AFTER options to prevent comments that were used in the source DDL from being included in the target DDL. By default, comments are not removed, so that they can be used for string substitution.

For more information about DDLOPTIONS, see *Reference for Oracle GoldenGate for Windows and UNIX*.

13.15 Replicating an IDENTIFIED BY Password

Use the DDLOPTIONS parameter with the DEFAULTUSERPASSWORDALIAS and REPLICATEPASSWORD | NOREPLICATEPASSWORD options to control how the password of a replicated {CREATE | ALTER} USER *name* IDENTIFIED BY *password* statement is handled. These options must be used together.

See the USEPASSWORDVERIFIERLEVEL option of DDLOPTIONS for important information about specifying the password verifier when Replicat operates against an Oracle 10*g* or 11*g* database.

For more information about DDLOPTIONS, see *Reference for Oracle GoldenGate for Windows and UNIX*.

> **Note:**
>
> Replication of CREATE | ALTER PROFILE will fail as the profile/password verification function must exist in the SYS schema. To replicate these DDLs successfully, password verification function must be created manually on both source/target(s) since DDL to SYS schema is excluded.

13.16 How DDL is Evaluated for Processing

The following explains how Oracle GoldenGate processes DDL statements on the source and target systems. It shows the order in which different criteria in the Oracle GoldenGate parameters are processed, and it explains the differences between how Extract and Replicat each process the DDL.

Extract

1. Extract captures a DDL statement.
2. Extract separates comments, if any, from the main statement.
3. Extract searches for the DDL parameter. (This example assumes it exists.)
4. Extract searches for the IGNOREREPLICATES parameter. If it is present, and if Replicat produced this DDL on this system, Extract ignores the DDL statement. (This example assumes no Replicat operations on this system.)
5. Extract determines whether the DDL statement is a RENAME. If so, the rename is flagged internally.
6. Extract gets the base object name and, if present, the derived object name.
7. If the statement is a RENAME, Extract changes it to ALTER TABLE RENAME.
8. Extract searches for the DDLOPTIONS REMOVECOMMENTS BEFORE parameter. If it is present, Extract removes the comments from the DDL statement, but stores them in case there is a DDL INCLUDE or DDL EXCLUDE clause that uses INSTR or INSTRCOMMENTS.
9. Extract determines the DDL scope: MAPPED, UNMAPPED or OTHER:
   - It is MAPPED if the operation and object types are supported for mapping, and the base object name and/or derived object name (if RENAME) is in a TABLE parameter.
   - It is UNMAPPED if the operation and object types are not supported for mapping, and the base object name and/or derived object name (if RENAME) is not in a TABLE parameter.
   - Otherwise the operation is identified as OTHER.

10. Extract checks the DDL parameter for INCLUDE and EXCLUDE clauses, and it evaluates the DDL parameter criteria in those clauses. All options must evaluate to TRUE in order for the INCLUDE or EXCLUDE to evaluate to TRUE. The following occurs:
    - If an EXCLUDE clause evaluates to TRUE, Extract discards the DDL statement and evaluates another DDL statement. In this case, the processing steps start over.
    - If an INCLUDE clause evaluates to TRUE, or if the DDL parameter does not have any INCLUDE or EXCLUDE clauses, Extract includes the DDL statement, and the processing logic continues.
11. Extract searches for a DDLSUBST parameter and evaluates the INCLUDE and EXCLUDE clauses. If the criteria in those clauses add up to TRUE, Extract performs string substitution. Extract evaluates the DDL statement against each DDLSUBST parameter in the parameter file. For all true DDLSUBST specifications, Extract performs string substitution in the order that the DDLSUBST parameters are listed in the file.
12. Now that DDLSUBT has been processed, Extract searches for the REMOVECOMMENTS AFTER parameter. If it is present, Extract removes the comments from the DDL statement.
13. Extract searches for DDLOPTIONS ADDTRANDATA. If it is present, and if the operation is CREATE TABLE, Extract issues the ALTER TABLE *name* ADD SUPPLEMENTAL LOG GROUP command on the table.
14. Extract writes the DDL statement to the trail.

Replicat

1. Replicat reads the DDL statement from the trail.
2. Replicat separates comments, if any, from the main statement.
3. Replicat searches for DDLOPTIONS REMOVECOMMENTS BEFORE. If it is present, Replicat removes the comments from the DDL statement.
4. Replicat evaluates the DDL synchronization scope to determine if the DDL qualifies for name mapping. Anything else is of OTHER scope.
5. Replicat evaluates the MAP statements in the parameter file. If the source base object name for this DDL (as read from the trail) appears in any of the MAP statements, the operation is marked as MAPPED in scope. Otherwise it is marked as UNMAPPED in scope.
6. Replicat replaces the source base object name with the base object name that is specified in the TARGET clause of the MAP statement.
7. If there is a derived object, Replicat searches for DDLOPTIONS MAPDERIVED. If it is present, Replicat replaces the source derived name with the target derived name from the MAP statement.
8. Replicat checks the DDL parameter for INCLUDE and EXCLUDE clauses, and it evaluates the DDL parameter criteria contained in them. All options must evaluate to TRUE in order for the INCLUDE or EXCLUDE to evaluate to TRUE. The following occurs:
    - If any EXCLUDE clause evaluates to TRUE, Replicat discards the DDL statement and starts evaluating another DDL statement. In this case, the processing steps start over.

- If any INCLUDE clause evaluates to TRUE, or if the DDL parameter does not have any INCLUDE or EXCLUDE clauses, Replicat includes the DDL statement, and the processing logic continues.

9. Replicat searches for the DDLSUBST parameter and evaluates the INCLUDE and EXCLUDE clauses. If the options in those clauses add up to TRUE, Replicat performs string substitution. Replicat evaluates the DDL statement against each DDLSUBST parameter in the parameter file. For all true DDLSUBST specifications, Replicat performs string substitution in the order that the DDLSUBST parameters are listed in the file.
10. Now that DDLSUBT has been processed, Replicat searches for the REMOVECOMMENTS AFTER parameter. If it is present, Replicat removes the comments from the DDL statement.
11. Replicat executes the DDL statement on the target database.
12. If there are no errors, Replicat processes the next DDL statement. If there are errors, Replicat performs the following steps.
13. Replicat analyzes the INCLUDE and EXCLUDE rules in the Replicat DDLERROR parameters in the order that they appear in the parameter file. If Replicat finds a rule for the error code, it applies the specified error handling; otherwise, it applies DEFAULT handling.
14. If the error handling does not enable the DDL statement to succeed, Replicat does one of the following: abends, ignores the operation, or discards it as specified in the rules.

> **Note:**
>
> If there are multiple targets for the same source in a MAP statement, the processing logic executes for each one.

13.17 Handling DDL Processing Errors

Use the DDLERROR parameter to handle errors on objects found by Extract for which metadata cannot be found, and for Replicat errors that occur when DDL is applied to the target database. With DDLERROR options, you can handle most errors in a default manner, for example to stop processing, and also handle other errors in a specific manner. You can use multiple instances of DDLERROR in the same parameter file to handle all errors that are anticipated.

For options and usage, see *Reference for Oracle GoldenGate for Windows and UNIX*.

13.18 Viewing DDL Report Information

By default, Oracle GoldenGate shows basic statistics about DDL at the end of the Extract and Replicat reports. To enable expanded DDL reporting, use the DDLOPTIONS parameter with the REPORT option. Expanded reporting includes the following information about DDL processing:

- A step-by-step history of the DDL operations that were processed by Oracle GoldenGate.

- The DDL filtering and processing parameters that are being used.

Expanded DDL report information increases the size of the report file, but it might be useful in certain situations, such as for troubleshooting or to determine when an `ADDTRANDATA` to add supplemental logging was applied.

To view a report, use the `VIEW REPORT` command in GGSCI.

```
VIEW REPORT group
```

- Viewing DDL Reporting in Replicat (page 13-32)
- Viewing DDL Reporting in Extract (page 13-33)
- Statistics in the Process Reports (page 13-33)

13.18.1 Viewing DDL Reporting in Replicat

The Replicat report lists:

- The entire syntax and source Oracle GoldenGate SCN of each DDL operation that Replicat processed from the trail. You can use the source SCN for tracking purposes, especially when there are restores from backup and Replicat is positioned backward in the trail.
- A subsequent entry that shows the scope of the operation (`MAPPED`, `UNMAPPED`, `OTHER`) and how object names were mapped in the target DDL statement, if applicable.
- Another entry that shows how processing criteria was applied.
- Additional entries that show whether the operation succeeded or failed, and whether or not Replicat applied error handling rules.

The following excerpt from a Replicat report illustrates a sequence of steps, including error handling:

```
2011-01-20 15:11:45  GGS INFO     2104  DDL found, operation [drop table
myTableTemp ], Source SCN [1186713.0].
 2011-01-20 15:11:45  GGS INFO     2100  DDL is of mapped scope, after mapping new
operation [drop table "QATEST2"."MYTABLETEMP" ].
 2011-01-20 15:11:45  GGS INFO     2100  DDL operation included [include objname
myTable*], optype [DROP], objtype [TABLE], objname [QATEST2.MYTABLETEMP].
 2011-01-20 15:11:45  GGS INFO     2100  Executing DDL operation.
 2011-01-20 15:11:48  GGS INFO     2105  DDL error ignored for next retry: error
code [942], filter [include objname myTableTemp], error text [ORA-00942: table or
view does not exist], retry [1].
 2011-01-20 15:11:48  GGS INFO     2100  Executing DDL operation , trying again due
to RETRYOP parameter.
 2011-01-20 15:11:51  GGS INFO     2105  DDL error ignored for next retry: error
code [942], filter [include objname myTableTemp], error text [ORA-00942: table or
view does not exist], retry [2].
 2011-01-20 15:11:51  GGS INFO     2100  Executing DDL operation, trying again due
to RETRYOP parameter.
 2011-01-20 15:11:54  GGS INFO     2105  DDL error ignored for next retry: error
code [942], filter [include objname myTableTemp], error text [ORA-00942: table or
view does not exist], retry [3].
 2011-01-20 15:11:54  GGS INFO     2100  Executing DDL operation, trying again due
to RETRYOP parameter.
 2011-01-20 15:11:54  GGS INFO     2105  DDL error ignored: error code [942], filter
[include objname myTableTemp], error text [ORA-00942: table or view does not exist].
```

13.18.2 Viewing DDL Reporting in Extract

The Extract report lists the following:

- The entire syntax of each captured DDL operation, the start and end SCN, the Oracle instance, the DDL sequence number (from the `SEQNO` column of the history table), and the size of the operation in bytes.
- A subsequent entry that shows how processing criteria was applied to the operation, for example string substitution or `INCLUDE` and `EXCLUDE` filtering.
- Another entry showing whether the operation was written to the trail or excluded.

The following, taken from an Extract report, shows an included operation and an excluded operation. There is a report message for the included operation, but not for the excluded one.

```
2011-01-20 15:11:41  GGS INFO     2100  DDL found, operation [create table myTable (
    myId number (10) not null,
    myNumber number,
    myString varchar2(100),
    myDate date,
    primary key (myId)
) ], start SCN [1186754], commit SCN [1186772] instance [test11g (1)], DDL seqno
[4134].

2011-01-20 15:11:41  GGS INFO     2100  DDL operation included [INCLUDE OBJNAME
myTable*], optype [CREATE], objtype [TABLE], objname [QATEST1.MYTABLE].

2011-01-20 15:11:41  GGS INFO     2100  DDL operation written to extract trail file.

2011-01-20 15:11:42  GGS INFO     2100  Successfully added TRAN DATA for table with
the key, table [QATEST1.MYTABLE], operation [ALTER TABLE "QATEST1"."MYTABLE" ADD
SUPPLEMENTAL LOG GROUP "GGS_MYTABLE_53475" (MYID) ALWAYS  /*
GOLDENGATE_DDL_REPLICATION */ ].

2011-01-20 15:11:43  GGS INFO     2100  DDL found, operation [create table
myTableTemp (
    vid varchar2(100),
    someDate date,
    primary key (vid)
) ], start SCN [1186777], commit SCN [1186795] instance [test11g (1)], DDL seqno
[4137].

2011-01-20 15:11:43  GGS INFO     2100  DDL operation excluded [EXCLUDE OBJNAME
myTableTemp OPTYPE CREATE], optype [CREATE], objtype [TABLE], objname
[QATEST1.MYTABLETEMP].
```

13.18.3 Statistics in the Process Reports

You can send current statistics for DDL processing to the Extract and Replicat reports by using the `SEND` command in GGSCI.

```
SEND {EXTRACT | REPLICAT} group REPORT
```

The statistics show totals for:

- All DDL operations
- Operations that are `MAPPED` in scope

- Operations that are `UNMAPPED` in scope
- Operations that are `OTHER` in scope
- Operations that were excluded (number of operations minus included ones)
- Errors (Replicat only)
- Retried errors (Replicat only)
- Discarded errors (Replicat only)
- Ignored operations (Replicat only)

13.19 Tracing DDL Processing

If you open a support case with Oracle GoldenGate Technical Support, you might be asked to turn on tracing. `TRACE` and `TRACE2` control DDL tracing. See *Reference for Oracle GoldenGate for Windows and UNIX*.

13.20 Using Tools that Support Trigger-Based DDL Capture

This section documents the additional tools available to support trigger-based capture.

- Tracing the DDL Trigger (page 13-34)
- Viewing Metadata in the DDL History Table (page 13-34)
- Handling DDL Trigger Errors (page 13-35)

13.20.1 Tracing the DDL Trigger

To trace the activity of the Oracle GoldenGate DDL trigger, use the following tools.

- `ggs_ddl_trace.log` trace file: Oracle GoldenGate creates a trace file in the `USER_DUMP_DEST` directory of Oracle. On RAC, each node has its own trace file that captures DDL tracing for that node. You can query the trace file as follows:

  ```
  select value from sys.v_$parameter where name = 'user_dump_dest';
  ```

- `ddl_tracelevel` script: Edit and run this script to set the trace level. A value of `None` generates no DDL tracing, except for fatal errors and installation logging. The default value of `0` generates minimal tracing information. A value of `1` or `2` generates a much larger amount of information in the trace file. Do not use `1` or `2` unless requested to do so by a Oracle GoldenGate Technical Support analyst as part of a support case.
- `ddl_cleartrace` script: Run this script on a regular schedule to prevent the trace file from consuming excessive disk space as it expands. It deletes the file, but Oracle GoldenGate will create another one. The DDL trigger stops writing to the trace file when the Oracle directory gets low on space, and then resumes writing when space is available again. This script is in the Oracle GoldenGate directory. Back up the trace file before running the script.

13.20.2 Viewing Metadata in the DDL History Table

Use the `DUMPDDL` command in GGSCI to view the information that is contained in the DDL history table. This information is stored in proprietary format, but you can export it in human-readable form to the screen or to a series of SQL tables that you can query.

The information in the DDL history table is the same as that used by the Extract process.

13.20.3 Handling DDL Trigger Errors

Use the `params.sql` non-executable script to handle failures of the Oracle GoldenGate DDL trigger in relation to whether the source DDL fails or succeeds. The `params.sql script` is in the root Oracle GoldenGate directory. The parameters to use are the following:

- **`ddl_fire_error_in_trigger`**: If set to `TRUE`, failures of the Oracle GoldenGate DDL trigger are raised with a Oracle GoldenGate error message and a database error message to the source end-user application. The source operations fails.

  If set to `FALSE`, no errors are raised, and a message is written to the trigger trace file in the Oracle GoldenGate directory. The source operation succeeds, but no DDL is replicated. The target application will eventually fail if subsequent data changes do not match the old target object structure. The default is `FALSE`.

- **`ddl_cause_error`**: If set to `TRUE`, tests the error response of the trigger by deliberately causing an error. To generate the error, Oracle GoldenGate attempts to `SELECT` zero rows without exception handling. Revert this flag to the default of `FALSE` after testing is done.

13.21 Using Edition-Based Redefinition

Oracle GoldenGate supports the use of Edition-based Redefinition (EBR) with Oracle Databases enabling you to upgrade the database component of an application while it is in use, thereby minimizing or eliminating down time.

Editions are non-schema objects that Editioned objects belong to. Editions can be thought of as owning editioned objects or as a namespace. Every database starts with one edition named, `ORA$BASE`; this includes upgraded databases. More than one edition can exist in a database and each can only have one child. For example, if you create three editions in succession, edition1, edition2, edition3, then edition1 is the parent of edition2 which is the parent of edition3. This is irrespective of the user or database session that creates them or which edition was current when the new one is created. When you create an edition, it inherits all the editioned objects of its parent. To use editions with Oracle GoldenGate, you must create them. For more information about creating and managing editions, see *Oracle Database Administrator's Guide* .

An object is considered editioned if it is an editionable type, it is created with the `EDITIONABLE` attribute, and the schema is enabled for editioning of that object type. When you create, alter, or drop an editioned object, the redo log will contain the name of the edition in which it belongs. In a container database, editions belong to the container and each container has its own default edition.

The `CREATE | DROP EDITION` DDLs are captured for all Extract configurations. They fall into the `OTHER` category and assigned an `OBJTYPE` option value of `EDITION`. The `OBJTYPE` option can be used for filtering, for example:

```
DDL EXCLUDE OBJTYPE EDITION
DDL EXCLUDE OBJTYPE EDITION OPTYPE CREATE
DDL EXCLUDE OBJTYPE EDITION OPTYPE DROP
DDL EXCLUDE OBJTYPE EDITION OPTYPE DROP ALLOWEMPTYOWNER OBJNAME edition_name
```

You must use the following syntax to exclude an edition from Extract or Replicat:

```
EXCLUDE OBJTYPE EDITION, ALLOWEMPTYOWNER OBJNAME edition_name
```

Editions fall into the `OTHER` category so no mapping is performed on the edition name. When applied, the `USE` permission is automatically granted to the Replicat user. Replicat will also perform a `grant use on edition name with grant option` to the original creating user if that user exists on the target database. Because editions are not mappable operations, they do not have owners so the standard `EXCLUDE` statement does not work.

The DDLs used to create or alter editions does not actually enable the user for editions, rather they enable the schema for editions. This is an important distinction because it means that the Replicat user does not need to be enabled for editions to apply DDLs to editioned objects. When Replicat applies a `CREATE EDITION` DDL, it grants the original creating user permission to `USE` it if the original user exists on the target database. For any unreplicated `CREATE EDITION` statements, you must issue a `USE WITH GRANT OPTION` grant to the Replicat user.

Whether or not an editionable objects becomes editioned is controlled by the schema it is applied in. Replicat switches its current session Edition before applying a DDL if the edition name attribute exists in the trail file and it is not empty.

Container database environments are supported for both Extract and Replicat. No additional configuration is necessary. The Replicat user's schema can not be enabled for editions if it is a common user. The Replicat user's schema does not need to be enabled for editions when applying DDLs to editioned objects in other schemas.

> **Note:**
>
> EBR support is limited to Integrated Dictionary; it is not supported when using a DDL trigger.

14
Creating Process Groups

This chapter contains instructions for creating Oracle GoldenGate process groups, collectively known as the "change-synchronization" processes. At minimum, you will create one primary Extract, one data pump, and one Replicat process group.
Topics:

14.1 Prerequisites to These Procedures

This chapter assumes you have installed Oracle GoldenGate, understand the different processing options available to you, and have performed the following prerequisite configuration steps before proceeding to configure Oracle GoldenGate process groups:

Establishing Oracle GoldenGate Credentials (page 4-1)

Preparing the Database for Oracle GoldenGate (page 3-1)

Configuring Capture in Integrated Mode (page 7-1)

Configuring Capture in Classic Mode (page 8-1)

Configuring Oracle GoldenGate Apply (page 9-1)

Configuring DDL Support (page 13-1) (to use DDL support)

14.2 Registering Extract with the Mining Database

If you are using Extract in integrated mode, you need to create a database logmining server to capture redo data. You do this from the GGSCI interface by registering the primary Extract process with the mining database. The creation of the logmining server extracts a snapshot of the source database in the redo stream of the source database. In a source multitenant container database, you register Extract with each of the pluggable databases that you want to include for capture.

> **WARNING:**
>
> Make certain that you know the earliest SCN of the log stream at which you want Extract to begin processing. Extract cannot have a starting SCN value that is lower than the first SCN that is specified when the underlying database capture process is created with the `REGISTER EXTRACT` command. You can use the `SCN` option

1. Log into the mining database then use the commands appropriate to your environment. The use of `DBLOGIN` always refers to the source database.

   Command for source database deployment:

   ```
   DBLOGIN USERIDALIAS alias
   ```

   Command for downstream mining database deployment:

   ```
   DBLOGIN USERIDALIAS alias
   MININGDBLOGIN USERIDALIAS alias2
   ```

   Where: *`alias`* specifies the alias of the database login credential that is assigned to Extract. This credential must exist in the Oracle GoldenGate credential store. For more information, see Establishing Oracle GoldenGate Credentials (page 4-1)For more information about `DBLOGIN`, see *Reference for Oracle GoldenGate for Windows and UNIX*. For more information about `MININGDBLOGIN`, see *Reference for Oracle GoldenGate for Windows and UNIX*.

2. Register the Extract process with the mining database.

   ```
   REGISTER EXTRACT group DATABASE [CONTAINER (container[, ...])] [SCN
   system_change_number]
   ```

   Where:

   - *`group`* is the name of the Extract group.
   - `CONTAINER (`*`container`*`[, ...])` specifies a pluggable database (PDB) within a multitenant container database, or a list of PDBs separated with commas. The specified PDBs must exist before the `REGISTER` command is executed. Extract will capture only from the PDBs that are listed in this command. For example, the following command registers PDBs `mypdb1` and `mypdb4`. Changes from any other PDBs in the multitenant container database are ignored by Oracle GoldenGate.

     ```
     REGISTER EXTRACT myextract DATABASE CONTAINER (mypdb1, mypdb4, mydb5)
     ```

     You can add or drop pluggable databases at a later date by stopping Extract, issuing a `DBLOGIN` command, and then issuing `REGISTER EXTRACT` with the `{ADD | DROP} CONTAINER` option of `DATABASE`. See *Reference for Oracle GoldenGate for Windows and UNIX* for more information about `REGISTER EXTRACT`.

     > **Note:**
     >
     > Adding `CONTAINERS` at particular SCN on an existing Extract is not supported.

- Registers Extract to begin capture at a specific SCN in the past. Without this option, capture begins from the time that `REGISTER EXTRACT` is issued. The specified SCN must correspond to the begin SCN of a dictionary build operation in a log file. You can issue the following query to find all valid SCN values:

```
SELECT first_change#
   FROM v$archived_log
   WHERE dictionary_begin = 'YES' AND
      standby_dest = 'NO' AND
      name IS NOT NULL AND
      status = 'A';
```

3. To register additional Extracts with a downstream database for the same source database, issue this `REGISTER` command.

   If you want to have more than one extract per source database, you can do that using the `SHARE` with `REGISTER EXTRACT` for better performance and metadata management. The specified `SCN` must correspond to the `SCN` where mining should begin in the archive logs.

```
REGISTER EXTRACT group DATABASE [CONTAINER (container[, ...])] [SCN
system_change_number] SHARE
```

> **Note:**
>
> The register process may take a few to several minutes to complete, even though the `REGISTER` command returns immediately.

14.3 Adding the Primary Extract

These steps add the primary Extract that captures change data.

1. If using downstream capture, set the RMAN archive log deletion policy to the following value in the source database:

```
CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON ALL STANDBY
```

   This must be done before you add the primary Extract.

2. Run GGSCI.

3. If using integrated capture, issue the `DBLOGIN` command.

```
DBLOGIN USERIDALIAS alias
```

   Where: *alias* specifies the alias of the database login credential that is assigned to Extract. This credential must exist in the Oracle GoldenGate credential store. For more information, see Establishing Oracle GoldenGate Credentials (page 4-1)

4. Issue the `ADD EXTRACT` command to add the primary Extract group.

```
ADD EXTRACT group name
{, TRANLOG | , INTEGRATED TRANLOG}
{, BEGIN {NOW | yyyy-mm-dd[ hh:mi:[ss[.ccccccc]]]} | SCN value}
[, THREADS n]
```

Where:

- *group name* is the name of the Extract group.
- TRANLOG specifies the transaction log as the data source; *for classic capture only*. See Example 14-1 (page 14-4).
- INTEGRATED TRANLOG specifies that Extract receives logical change records through a database logmining server; *for integrated capture only*. See Example 14-2 (page 14-4). Before issuing ADD EXTRACT with this option, make certain you logged in to the database with the DBLOGIN command and that you registered this Extract with the database. See Registering Extract with the Mining Database (page 14-1) for more information.
- BEGIN specifies to begin capturing data as of a specific *time*:
  - NOW starts at the first record that is time stamped at the same time that ADD EXTRACT is issued.
  - *yyyy-mm-dd*[*hh:mi:*[*ss*[*.cccccc*]]] starts at an explicit timestamp. Logs from this timestamp must be available. For Extract in integrated mode, the timestamp value must be greater than the timestamp at which the Extract was registered with the database.
  - SCN *value* starts Extract at the transaction in the redo log that has the specified Oracle system change number (SCN). For Extract in integrated mode, the SCN value must be greater than the SCN at which the Extract was registered with the database. See Registering Extract with the Mining Database (page 14-1) for more information.
- THREADS *n* is required in classic capture mode for Oracle Real Application Cluster (RAC), to specify the number of redo log threads being used by the cluster. Extract reads and coordinates each thread to maintain transactional consistency. Not required for integrated capture.

> **Note:**
>
> Additional options are available. See *Reference for Oracle GoldenGate for Windows and UNIX*.

Example 14-1 Classic capture with timestamp start point

```
ADD EXTRACT finance, TRANLOG, BEGIN 2011-01-01 12:00:00.000000
```

Example 14-2 Integrated capture with timestamp start point

```
DBLOGIN USERIDALIAS myalias
ADD EXTRACT finance, INTEGRATED TRANLOG, BEGIN NOW
```

14.4 Add the Local Trail

These steps add the local trail to which the primary Extract writes captured data.

In GGSCI on the source system, issue the ADD EXTTRAIL command:

```
ADD EXTTRAIL pathname, EXTRACT group name
```

Where:

- `EXTTRAIL` specifies that the trail is to be created on the local system.
- *`pathname`* is the relative or fully qualified name of the trail, including the two-character name.
- `EXTRACT` *`group name`* is the name of the primary Extract group.

> **Note:**
>
> Oracle GoldenGate creates this trail automatically during processing.

Example 14-3

```
ADD EXTTRAIL /ggs/dirdat/lt, EXTRACT finance
```

14.5 Add the Data Pump Extract Group

These steps add the data pump that reads the local trail and sends the data to the target.

In GGSCI on the source system, issue the `ADD EXTRACT` command.

`ADD EXTRACT` *`group name,`* `EXTTRAILSOURCE` *`trail name`*

Where:

- group name is the name of the Extract group.
- `EXTTRAILSOURCE` *`trail name`* is the relative or fully qualified name of the local trail.

Example 14-4

```
ADD EXTRACT financep, EXTTRAILSOURCE c:\ggs\dirdat\lt
```

14.6 Add the Remote Trail

These steps add the remote trail. Although it is read by Replicat, this trail must be associated with the data pump, so it must be added on the source system, not the target.

In GGSCI on the source system, issue the following command:

`ADD RMTTRAIL` *`pathname,`* `EXTRACT` *`group name`*

Where:

- `RMTTRAIL` specifies that the trail is to be created on the target system.
- pathname is the relative or fully qualified name of the trail, including the two-character name.
- `EXTRACT` *`group name`* is the name of the data-pump Extract group.

> **Note:**
>
> Oracle GoldenGate creates this trail automatically during processing.

Example 14-5

```
ADD RMTTRAIL /ggs/dirdat/rt, EXTRACT financep
```

14.7 Add the Replicat Group

These steps add the Replicat group that reads the remote trail and applies the data changes to the target Oracle Database.

1. Run GGSCI on the target system.
2. If using integrated Replicat, issue the `DBLOGIN` command to log into the database from GGSCI.

   ```
   DBLOGIN USERIDALIAS alias
   ```

   Where: *`alias`* specifies the alias of the database login credential that is assigned to Replicat. This credential must exist in the Oracle GoldenGate credential store. For more information, see Establishing Oracle GoldenGate Credentials (page 4-1)
3. Issue the `ADD REPLICAT` command with the following syntax.

   ```
   ADD REPLICAT group name, [INTEGRATED,] EXTTRAIL pathname
   ```

   Where:

   - *`group name`* is the name of the Replicat group.
   - `INTEGRATED` creates an integrated Replicat group.
   - `EXTTRAIL` *`pathname`* is the relative or fully qualified name of the remote trail, including the two-character name.

     For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.

Example 14-6 Adds a Nonintegrated Replicat

```
ADD REPLICAT financer, EXTTRAIL c:\ggs\dirdat\rt
```

Example 14-7 Adds an Integrated Replicat

```
ADD REPLICAT financer, INTEGRATED, EXTTRAIL c:\ggs\dirdat\rt
```

15

Instantiating Oracle GoldenGate Replication

This chapter contains instructions for configuring and performing an instantiation of the replication environment to establish and maintain a synchronized state between two or more databases. In a synchronized state, the source and target objects contain identical or appropriately corresponding values, depending on whether any conversion or transformation is performed on the data before applying it to the target objects.

Topics:

15.1 Overview of the Instantiation Process

In the instantiation procedure, you make a copy of the source data and load the copy to the target database. The initial load captures a point-in-time snapshot of the data, while Oracle GoldenGate maintains that consistency by applying transactional changes that occur while the static data is being loaded. After instantiation is complete, Oracle GoldenGate maintains the synchronized state throughout ongoing transactional changes.

When you instantiate Oracle GoldenGate processing, it is recommended that you do so first in a test environment before deploying live on your production machines. This is especially important in an active-active or high availability configuration, where trusted source data may be touched by the replication processes. Testing enables you to find and resolve any configuration mistakes or data issues without the need to interrupt user activity for re-loads on the target or other troubleshooting activities. Testing also ensures that your instantiation process is configured properly. Parameter files can be copied to the production equipment after successful testing, and then you can perform a predictable instantiation with production data.

15.2 Satisfying Prerequisites for Instantiation

The following steps must be taken before starting any Oracle GoldenGate processes or native database load processes.

- Adding Collision Handling (page 15-2)
- Preparing the Target Tables (page 15-3)
- Configuring and Adding Change Synchronization Groups (page 15-2)
- Disabling DDL Processing (page 15-2)
- Adding Collision Handling (page 15-2)
- Preparing the Target Tables (page 15-3)

15.2.1 Configuring and Adding Change Synchronization Groups

To perform an instantiation of the target database and the replication environment, the online change capture and apply groups must exist and be properly configured. See:

Configuring Capture in Integrated Mode (page 7-1)

Configuring Capture in Classic Mode (page 8-1)

Configuring Oracle GoldenGate Apply (page 9-1)

Creating Process Groups (page 14-1)

15.2.2 Disabling DDL Processing

You must disable DDL activities before performing an instantiation. You can resume DDL after the instantiation is finished. See Disabling DDL Processing Temporarily (page 16-1) for instructions.

15.2.3 Adding Collision Handling

This prerequisite applies to the following instantiation methods:

- Configuring a Direct Bulk Load to SQL*Loader (page 15-4)
- Configuring a Load from an Input File to SQL*Loader (page 15-7).

This prerequisite *does not* apply to the instantiation method described in Configuring a Load with an Oracle Data Pump (page 15-3).

If the source database will remain active during one of those initial load methods, collision-handling logic must be added to the Replicat parameter file. This logic handles conflicts that occur because static data is being loaded to the target tables while Oracle GoldenGate replicates transactional changes to those tables.

To handle collisions, add the `HANDLECOLLISIONS` parameter to the Replicat parameter file to resolve these collisions:

- `INSERT` operations for which the row already exists
- `UPDATE` and `DELETE` operations for which the row does not exist

`HANDLECOLLISIONS` should be removed from the Replicat parameter file at the end of the instantiation steps (as prompted in the instructions). For more information about `HANDLECOLLISIONS`, see *Reference for Oracle GoldenGate for Windows and UNIX*.

To use the `HANDLECOLLISIONS` function to reconcile incremental data changes with the load, each target table must have a primary or unique key. If you cannot create a key through your application, use the `KEYCOLS` option of the `TABLE` and `MAP` parameters to

specify columns as a substitute key for Oracle GoldenGate to use. If you cannot create keys, the affected source table must be quiesced for the load. See *Reference for Oracle GoldenGate for Windows and UNIX* for more information about `KEYCOLS`.

15.2.4 Preparing the Target Tables

The following are suggestions that can make the load go faster and help you to avoid errors.

- **Data**: Make certain that the target tables are empty. Otherwise, there may be duplicate-row errors or conflicts between existing rows and rows that are being loaded.
- **Indexes**: Remove indexes from the target tables. Indexes are not necessary for the inserts performed by the initial load process and will slow it down. You can add back the indexes after the load is finished.

15.3 Configuring the Initial Load

Oracle GoldenGate supports the following load methods specifically for Oracle:

- Configuring a Load with an Oracle Data Pump (page 15-3)
- Configuring a Direct Bulk Load to SQL*Loader (page 15-4)
- Configuring a Load from an Input File to SQL*Loader (page 15-7)

Select a method and follow its configuration steps to create the load processes and parameter files.

Some of the parameters that you use in a change-synchronization parameter file also are required in an initial-load Extract and initial-load Replicat parameter file. To take advantage of the commonalities, you can use any of the following methods:

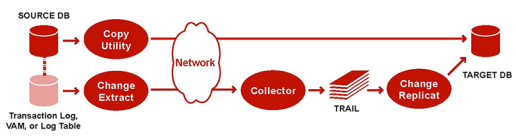
- Copy common parameters from one parameter file to another.
- Store the common parameters in a central file and use the `OBEY` parameter in each parameter file to retrieve them.
- Create an Oracle GoldenGate macro for the common parameters and then call the macro from each parameter file with the `MACRO` parameter.

For more information about working with parameter files, see *Administering Oracle GoldenGate for Windows and UNIX*.

- Configuring a Load with an Oracle Data Pump (page 15-3)
- Configuring a Direct Bulk Load to SQL*Loader (page 15-4)
- Configuring a Load from an Input File to SQL*Loader (page 15-7)

15.3.1 Configuring a Load with an Oracle Data Pump

Figure 15-1 Configuring a Load with the Oracle Data Pump Utility



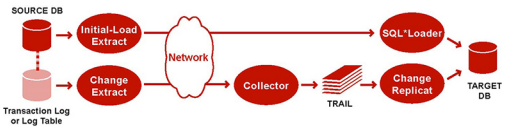
This method uses the Oracle Data Pump utility to establish the target data. You start Extract, the data pumps, and Replicat at the SCN at which the copy stopped. Transactions that were included in the copy are skipped to avoid collisions from integrity violations. From the process start point, Oracle GoldenGate maintains data synchronization.

No initial-load Oracle GoldenGate processes are required for this method.

15.3.2 Configuring a Direct Bulk Load to SQL*Loader

Figure 15-2 (page 15-4) shows configuring a direct bulk load to SQL*Loader.

Figure 15-2 Configuring a Direct Bulk Load to SQL*Loader



With this method, you configure and run an Oracle GoldenGate initial-load Extract to extract complete source records and send them directly to an initial-load Replicat task. The initial-load Replicat task communicates with SQL*Loader to load data as a direct-path bulk load. Data mapping and transformation can be done by either the initial-load Extract or initial-load Replicat, or both. During the load, the change-synchronization groups that you configured in Configuring Capture in Integrated Mode (page 7-1) or Configuring Capture in Classic Mode (page 8-1) and Configuring Oracle GoldenGate Apply (page 9-1) replicate incremental changes, which are then reconciled with the results of the load.

Limitations:

- This method does not support extraction of `LOB` or `LONG` data. As an alternative, see Performing Instantiation From an Input File to SQL*Loader (page 15-11).
- This method does not support materialized views that contain `LOBS`, regardless of their size. It also does not support data encryption.

To Configure a Direct Bulk Load to SQL*Loader

1. Grant `LOCK ANY TABLE` to the Replicat database user on the target Oracle Database.
2. On the source and target systems, run GGSCI.
3. Start Manager on both systems.

   ```
   START MANAGER
   ```

4. On the source system, create the initial-load Extract.

   ```
   ADD EXTRACT initial-load_Extract, SOURCEISTABLE
   ```

   Where:

   - *`initial-load_Extract`* is the name of the initial-load Extract, up to eight characters.
   - `SOURCEISTABLE` directs Extract to read complete records directly from the source tables.

5. On the source system, create the initial-load Extract parameter file.

   ```
   EDIT PARAMS initial-load_Extract
   ```

6. Enter the initial-load Extract parameters in the order shown, starting a new line for each parameter statement. This example shows a three-part table name associated with a multitenant container database. Refer to Table 15-1 (page 15-5) for descriptions.

   ```
   EXTRACT initext
   USERIDALIAS tiger1
   RMTHOST fin1, MGRPORT 7809 ENCRYPT AES192, KEYNAME securekey2
   RMTTASK replicat, GROUP initrep
   TABLE hq.hr.*;
   ```

Table 15-1 Initial-load Extract Parameters to Direct Bulk Load to SQL*Loader

Parameter	Description
`EXTRACT initial-load_Extract`	Specifies the name of the initial-load Extract, as stated with `ADD EXTRACT`. See *Reference for Oracle GoldenGate for Windows and UNIX*.
`USERIDALIAS alias`	Specifies the alias of the database login credential that is assigned to Extract. This credential must exist in the Oracle GoldenGate credential store. For more information, see Establishing Oracle GoldenGate Credentials (page 4-1)
`RMTHOST hostname, MGRPORT portnumber[, ENCRYPT algorithm KEYNAME keyname]`	Specifies the target system, the port where Manager is running, and optional encryption of data across TCP/IP. See *Reference for Oracle GoldenGate for Windows and UNIX*.
`RMTTASK REPLICAT, GROUP initial-load_Replicat`	Specifies the process type (must be `REPLICAT`) and the name of the initial-load Replicat. Directs Manager on the target system to dynamically start the initial-load Replicat as a one-time task. See *Reference for Oracle GoldenGate for Windows and UNIX*.
`TABLE [container.]schema.table;`	Specifies the tables to capture. • `container` is the name of the pluggable database, if this is a multitenant container database. You can use the `SOURCECATALOG` parameter to specify a default pluggable database instead of using three-part names. • `schema` is the schema name. • `table` is the table name. See *Administering Oracle GoldenGate for Windows and UNIX* for important information about specifying object names.

7. Save and close the file.

8. On the target system, create the initial-load Replicat.

   ```
   ADD REPLICAT initial-load Replicat, SPECIALRUN
   ```

   Where:

   - `initial-load Replicat` is the name of the initial-load Replicat task.

- `SPECIALRUN` identifies the initial-load Replicat as a one-time task, not a continuous process.

9. On the target system, create the initial-load Replicat parameter file.

```
EDIT PARAMS initial-load Replicat
```

10. Enter the initial-load Replicat parameters in the order shown, starting a new line for each parameter statement. This example shows a three-part source table name associated with a multitenant container database. See Table 15-2 (page 15-6) for descriptions.

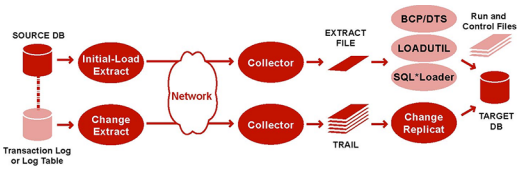
```
REPLICAT initrep
USERIDALIAS tiger2
BULKLOAD
ASSUMETARGETDEFS
MAP hq.hr.*, TARGET hr2.*;
```

Table 15-2 Initial-load Replicat Parameters to Direct Bulk Load to SQL*Loader

Parameter	Description
`REPLICAT initial-load Replicat`	Specifies the name of the initial-load Replicat task, as stated with `ADD REPLICAT`. See *Reference for Oracle GoldenGate for Windows and UNIX*.
`USERIDALIAS alias`	Specifies the alias of the database login credential that is assigned to Replicat. This credential must exist in the Oracle GoldenGate credential store. For more information, see Establishing Oracle GoldenGate Credentials (page 4-1)
`BULKLOAD`	Directs Replicat to interface directly with the Oracle SQL*Loader interface. See *Reference for Oracle GoldenGate for Windows and UNIX*
`ASSUMETARGETDEFS`	Assumes the source and target tables are identical, including semantics. If source and target definitions are different, you must create and specify a source-definitions file that both the change-synchronization and initial-load processes will use. See *Reference for Oracle GoldenGate for Windows and UNIX*. For more information about data-definitions files, see Administering Oracle GoldenGate for Windows and UNIX.
`MAP [container.]schema.table, TARGET schema.table;`	Specifies a relationship between a source and target table or tables. • If the source is a multitenant container database, *container* is the name of the pluggable database that contains the source objects specified with this `MAP` statement. You can use the `SOURCECATALOG` parameter to specify a default source pluggable database instead of using three-part names. • *schema* is the schema name. • *table* is the table name. See *Administering Oracle GoldenGate for Windows and UNIX* for important information about specifying object names.

15.3.3 Configuring a Load from an Input File to SQL*Loader

Figure 15-3 Configuring a Load from an Input File to SQL*Loader



With this method, an initial-load Extract extracts source records from the source tables and writes them to an extract file in external ASCII format. The files are read by SQL*Loader. During the load, the change-synchronization groups that you configured in Chapter 4 replicate incremental changes, which are then reconciled with the results of the load. As part of the load procedure, Oracle GoldenGate uses the initial-load Replicat to create run and control files required by the database utility. Any data transformation must be performed by the initial-load Extract on the source system because the control files are generated dynamically and cannot be pre-configured with transformation rules.

To Configure a Load from File to SQL*Loader

1. On the source and target systems, run GGSCI.
2. Start Manager on both systems.

   ```
   START MANAGER
   ```

3. On the source system, create the initial-load Extract parameter file.

   ```
   EDIT PARAMS initial-load Extract
   ```

4. Enter the initial-load Extract parameters in the order shown, starting a new line for each parameter statement. This example shows a three-part table name associated with a multitenant container database. Refer to Table 15-3 (page 15-7) for descriptions.

   ```
   SOURCEISTABLE
   USERIDALIAS tiger1
   RMTHOST fin1, MGRPORT 7809 ENCRYPT AES192, KEYNAME securekey2
   ENCRYPTTRAIL AES192
   FORMATASCII, SQLLOADER
   RMTFILE /ggs/dirdat/ie
   TABLE hq.hr.*;
   ```

Table 15-3 Initial-load Extract Parameters to Load From File to SQL*Loader

Parameter	Description
SOURCEISTABLE	Designates Extract as an initial load process that extracts records directly from the source tables. See *Reference for Oracle GoldenGate for Windows and UNIX*.

Table 15-3 (Cont.) Initial-load Extract Parameters to Load From File to SQL*Loader

Parameter	Description
`USERIDALIAS alias`	Specifies the alias of the database login credential that is assigned to Extract. This credential must exist in the Oracle GoldenGate credential store. For more information, see Establishing Oracle GoldenGate Credentials (page 4-1)
`RMTHOST hostname, MGRPORT portnumber[, ENCRYPT algorithm KEYNAME keyname]`	Specifies the target system, the port where Manager is running, and optional encryption of data across TCP/IP. See *Reference for Oracle GoldenGate for Windows and UNIX*.
`ENCRYPTTRAIL algorithm`	Encrypts the data in the remote file. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.
`FORMATASCII, SQLLOADER`	Produces a fixed-length, ASCII-formatted remote file that is compatible with SQL*Loader. This parameter must be listed before `RMTFILE`. See *Reference for Oracle GoldenGate for Windows and UNIX*.
`RMTFILE path`	Specifies the absolute or full path name of an extract file that Extract creates and to which it writes the load data. See *Reference for Oracle GoldenGate for Windows and UNIX*.
`TABLE [container.]schema.table;`	Specifies the tables to capture. • *container* is the name of the pluggable database, if this is a multitenant container database. You can use the `SOURCECATALOG` parameter to specify a default pluggable database instead of using three-part names. • *schema* is the schema name. • *table* is the table name. See *Administering Oracle GoldenGate for Windows and UNIX* for important information about specifying object names.

5. Save and close the parameter file.
6. On the target system, create the initial-load Replicat parameter file.

```
EDIT PARAMS initial-load Replicat
```

7. Enter the initial-load Replicat parameters in the order shown, starting a new line for each parameter statement. This example shows a three-part source table name associated with a multitenant container database. See Table 15-4 (page 15-8) for descriptions.

```
GENLOADFILES sqlldr.tpl
USERIDALIAS tiger2
EXTFILE /ggs/dirdat/ie
ASSUMETARGETDEFS
MAP hq.hr.*, TARGET hr2.*;
```

Table 15-4 Initial-load Replicat parameters to Load from File to SQL*Loader

Parameter	Description
`GENLOADFILES template`	Generates run and control files for the database utility. See *Reference for Oracle GoldenGate for Windows and UNIX*.

Table 15-4 (Cont.) Initial-load Replicat parameters to Load from File to SQL*Loader

Parameter	Description
`USERIDALIAS alias`	Specifies the alias of the database login credential of the user that is assigned to Replicat. This credential must exist in the Oracle GoldenGate credential store. For more information, see Establishing Oracle GoldenGate Credentials (page 4-1)
`EXTFILE path`	Specifies the extract file that you specified with the Extract parameter `RMTFILE`. See *Reference for Oracle GoldenGate for Windows and UNIX*.
`ASSUMETARGETDEFS`	Assumes the source and target tables are identical, including semantics. If source and target definitions are different, you must create and specify a source-definitions file that both the change-synchronization and initial-load processes will use. See *Reference for Oracle GoldenGate for Windows and UNIX*. For more information about data-definitions files, see *Administering Oracle GoldenGate for Windows and UNIX*.
`MAP [container.]schema.table, TARGET schema.table;`	Specifies a relationship between a source and target table or tables. • If the source is a multitenant container database, *container* is the name of the pluggable database that contains the source objects specified with this `MAP` statement. You can use the `SOURCECATALOG` parameter to specify a default source pluggable database instead of using three-part names. • *schema* is the schema name. • *table* is the table name. See *Administering Oracle GoldenGate for Windows and UNIX* for important information about specifying object names.

8. Save and close the parameter file.

15.4 Performing the Target Instantiation

This procedure instantiates the target tables while Oracle GoldenGate captures ongoing transactional changes on the source and stores them until they can be applied on the target. By the time you perform the instantiation of the target tables, the entire Oracle GoldenGate environment should be configured for change capture and delivery, as should the initial-load processes if using Oracle GoldenGate as an initial-load utility.

> **Note:**
>
> The first time that Extract starts in a new Oracle GoldenGate configuration, any open source transactions will be skipped. Only transactions that begin after Extract starts are captured.

- Performing Instantiation with Oracle Data Pump (page 15-10)
- Performing Instantiation with Direct Bulk Load to SQL*Loader (page 15-10)

- Performing Instantiation From an Input File to SQL*Loader (page 15-11)

15.4.1 Performing Instantiation with Oracle Data Pump

To perform instantiation with Oracle Data Pump, see My Oracle Support document 1276058.1. To obtain this document, do the following:

1. Go to `http://support.oracle.com`.
2. Under Sign In, select your language and then log in with your Oracle Single Sign-On (SSO).
3. On the Dashboard, expand the Knowledge Base heading.
4. Under Enter Search Terms, paste or type the document ID of `1276058.1` and then click **Search**.
5. In the search results, select **Oracle GoldenGate Best Practices: Instantiation from an Oracle Source Database [Article ID 1276058.1]**.
6. Click the link under Attachments to open the article.

15.4.2 Performing Instantiation with Direct Bulk Load to SQL*Loader

1. On the source system, run GGSCI.
2. Start the primary change-capture Extract group.

   ```
   START EXTRACT group
   ```

3. Start the data-pump Extract group.

   ```
   START EXTRACT data_pump
   ```

4. If replicating sequence values:
   - Issue the `DBLOGIN` command with the alias of a user in the credential store who has `EXECUTE` privilege on `update.Sequence`.

     ```
     DBLOGIN USERIDALIAS alias
     ```

   - Issue the following command to update each source sequence and generate redo. From the redo, Replicat performs initial synchronization of the sequences on the target. For more information about this command, see *Reference for Oracle GoldenGate for Windows and UNIX*.

     ```
     FLUSH SEQUENCE [container.]schema.sequence
     ```

5. Start the initial-load Extract.

   ```
   START EXTRACT initial-load_Extract
   ```

   > **WARNING:**
   >
   > Do not start the initial-load Replicat. The Manager process starts it automatically and terminates it when the load is finished.

6. On the target system, run GGSCI.
7. Issue the `VIEW REPORT` command to determine when the initial load to SQL*Loader is finished.

```
VIEW REPORT initial-load_Extract
```

8. When the load is finished, start the change-data Replicat group.

   ```
   START REPLICAT group
   ```

9. Issue the `INFO REPLICAT` command, and continue to issue it until it shows that Replicat posted all of the change data that was generated during the initial load. For example, if the initial-load Extract stopped at 12:05, make sure Replicat posted data up to that time.

   ```
   INFO REPLICAT group
   ```

10. Turn off `HANDLECOLLISIONS` for the change-delivery Replicat to disable initial-load error handling.

    ```
    SEND REPLICAT group, NOHANDLECOLLISIONS
    ```

11. Edit the change-delivery Replicat parameter file to remove the `HANDLECOLLISIONS` parameter.

    ```
    EDIT PARAMS group
    ```

12. Save and close the parameter file.

From this point forward, Oracle GoldenGate continues to synchronize data changes.

15.4.3 Performing Instantiation From an Input File to SQL*Loader

> **Note:**
>
> The SQL*Loader method is not recommended if the data has multibyte characters, especially when the character set of the operating system is different from the database character set.

1. On the source system, run GGSCI.
2. Start the primary change-capture Extract group.

   ```
   START EXTRACT group
   ```

3. Start the data-pump Extract group.

   ```
   START EXTRACT data_pump
   ```

4. If replicating sequence values:
   - Issue the `DBLOGIN` command with the alias of a user in the credential store who has `EXECUTE` privilege on `update.Sequence`.

     ```
     DBLOGIN USERIDALIAS alias
     ```

   - Issue the following command to update each source sequence and generate redo. From the redo, Replicat performs initial synchronization of the sequences on the target. For more information about this command, see *Reference for Oracle GoldenGate for Windows and UNIX*.

     ```
     FLUSH SEQUENCE [container.]schema.sequence
     ```

5. From the Oracle GoldenGate installation directory on the source system, start the initial-load Extract from the command line of the operating system (not GGSCI).

UNIX and Linux:

```
$ /OGG_directory/extract paramfile dirprm/initial-load_Extract.prm reportfile
path
```

Windows:

```
C:\> OGG_directory\extract paramfile dirprm\initial-load_Extract.prm
reportfile path
```

Where: *initial-load_Extract* is the name of the initial-load Extract and *path* is the relative or fully qualified path where you want the Extract report file to be created.

6. Wait until the initial extraction from the source is finished. Verify its progress and results by viewing the Extract report file from the command line.
7. On the target system, start the initial-load Replicat.

   UNIX and Linux:

   ```
   $ /OGG directory/replicat paramfile dirprm/initial-load_Replicat name.prm
   reportfile path
   ```

   Windows:

   ```
   C:\> OGG directory\replicat paramfile dirprm\initial-load_Replicat.prm
   reportfile path
   ```

   Where: `initial-load Extract` is the name of the initial-load Replicat and *path* is the relative or fully qualified path where you want the Replicat report file to be created.
8. When the initial-load Replicat stops, verify its results by viewing the Replicat report file from the command line.
9. Using the ASCII-formatted file and the run and control files that the initial-load Replicat created, load the data with SQL*Loader.
10. When the load is finished, start the change-delivery Replicat group.

    ```
    START REPLICAT group
    ```

11. Issue the `INFO REPLICAT` command, and continue to issue it until it shows that Replicat posted all of the change data that was generated during the initial load. For example, if the initial-load Extract stopped at 12:05, make sure Replicat posted data up to that time.

    ```
    INFO REPLICAT group
    ```

12. Turn off `HANDLECOLLISIONS` for the change-delivery Replicat to disable initial-load error handling.

    ```
    SEND REPLICAT group, NOHANDLECOLLISIONS
    ```

13. Edit the change-delivery Replicat parameter file to remove the `HANDLECOLLISIONS` parameter.

    ```
    EDIT PARAMS group
    ```

14. Save and close the parameter file.

From this point forward, Oracle GoldenGate continues to synchronize data changes.

15.5 Monitoring and Controlling Processing After the Instantiation

After the target is instantiated and replication is in effect, you can control processes and view the overall health of the replication environment.

- To control processes, see *Administering Oracle GoldenGate for Windows and UNIX*.
- To ensure that all processes are running properly and that errors are being handled according to your error handling rules, see *Administering Oracle GoldenGate for Windows and UNIX*. Oracle GoldenGate provides commands and logs to view process status, lag, warnings, and other information.

Additionally, if you configured Replicat in integrated mode, you can use the `STATS REPLICAT` command to view statistics on the number of transactions that are applied in integrated mode as compared to those that are applied in direct apply mode.

```
STATS REPLICAT group
```

The output of this command shows the number of transactions applied, the number of transactions that were redirected to direct apply, and the direct transaction ratio, among other statistics. The statistics help you determine whether integrated Replicat is performing as intended. If the environment is satisfactory and there is a high ratio of direct apply operations, consider using nonintegrated Replicat. You can configure parallelism with nonintegrated Replicat.

> **Note:**
>
> To ensure realistic statistics, view apply statistics only after you are certain that the Oracle GoldenGate environment is well established, that configuration errors are resolved, and that any anticipated processing errors are being handled properly.

You can also view runtime statistics for integrated Replicat in the `V$`views for each of the inbound server components. For more information, see *Oracle Database Reference*.

- The reader statistics are recorded in `V$GG_APPLY_READER` and include statistics on number of messages read, memory used, and dependency counts.
- The apply coordinator statistics are recorded in `V$GG_APPLY_COORDINATOR` and record statistics at the transaction level.
- The apply server statistics are recorded in `V$GG_APPLY_SERVER`. This view records information for each of the apply server processes (controlled by `parallelism` and `max_parallelism` parameters) as separate rows. The statistics for each apply server are identified by the `SERVER_ID` column. If a `SERVER_ID` of `0` exists, this represents an aggregate of any apply servers that exited because the workload was reduced.
- Statistics about the number of messages received by the database from Replicat are recorded in the `V$GG_APPLY_RECEIVER` table.

Additionally, you can query the following views. For more information about this view, see Oracle Database Reference.

- `V$GOLDENGATE_TABLE_STATS` to see statistics for DML and collisions that occurred for each replicated table that the inbound server processed.
- `V$GOLDENGATE_TRANSACTION` to see information about transactions that are being processed by Oracle GoldenGate inbound servers.

15.6 Verifying Synchronization

To verify that the source and target data are synchronized, you can use the Oracle GoldenGate Veridata product or use your own scripts to select and compare source and target data.

15.7 Backing up the Oracle GoldenGate Environment

After you start Oracle GoldenGate processing, an effective backup routine is critical to preserving the state of processing in the event of a failure. Unless the Oracle GoldenGate working files can be restored, the entire replication environment must be re-instantiated, complete with new initial loads.

As a best practice, include the entire Oracle GoldenGate home installation in your backup routines. There are too many critical sub-directories, as well as files and programs at the root of the directory, to keep track of separately. In any event, the most critical files are those that consume the vast majority of backup space, and therefore it makes sense just to back up the entire installation directory for fast, simple recovery.

16

Managing the DDL Replication Environment

This chapter contains instructions for making changes to the database environment or the Oracle GoldenGate environment when the Oracle GoldenGate DDL trigger is being used to support DDL replication. See Installing Trigger-Based DDL Capture (page D-1) for more information about the DDL objects.
For instructions on configuring Oracle GoldenGate DDL support, see Configuring DDL Support (page 13-1).

> **Note:**
>
> This chapter is only relevant for classic capture mode or integrated capture mode in which trigger-based DDL capture is being used.

Topics:

16.1 Disabling DDL Processing Temporarily

You must disable DDL activities before performing an instantiation or other tasks, if directed. You can resume DDL processing after the task is finished.

1. Disable user DDL operations on the source database.
2. If there are previous DDL replication processes that are still active, make certain that the last executed DDL operation was applied to the target before stopping

those processes, so that the load data is applied to objects that have the correct metadata.

3. Comment out the `DDL` parameter in the Extract and Replicat parameter files that you configured for the new Oracle GoldenGate environment. Comment out any other parameters that support DDL.
4. Disable the Oracle GoldenGate DDL trigger, if one is in use. See Enabling and Disabling the DDL Trigger (page 16-2).

16.2 Enabling and Disabling the DDL Trigger

You can enable and disable the trigger that captures DDL operations without making any configuration changes within Oracle GoldenGate. The following scripts control the DDL trigger.

- `ddl_disable`: Disables the trigger. No further DDL operations are captured or replicated after you disable the trigger.
- `ddl_enable`: Enables the trigger. When you enable the trigger, Oracle GoldenGate starts capturing current DDL changes, but does not capture DDL that was generated while the trigger was disabled.

Before running these scripts, disable all sessions that ever issued DDL, including those of the Oracle GoldenGate processes, SQL*Plus, business applications, and any other software that uses Oracle. Otherwise the database might generate an ORA-04021 error. Do not use these scripts if you intend to maintain consistent DDL on the source and target systems.

16.3 Maintaining the DDL Marker Table

You can purge rows from the marker table at any time. It does not keep DDL history. To purge the marker table, use the Manager parameter `PURGEMARKERHISTORY`. Manager gets the name of the marker table from one of the following:

1. The name given with the `MARKERTABLE` parameter in the `GLOBALS` file, if specified.
2. The default name of `GGS_MARKER`.

`PURGEMARKERHISTORY` provides options to specify maximum and minimum lengths of time to keep a row, based on the last modification date. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.

16.4 Deleting the DDL Marker Table

Do not delete the DDL marker table unless you want to discontinue synchronizing DDL. The marker table and the DDL trigger are interdependent. An attempt to drop the marker table fails if the DDL trigger is enabled. This is a safety measure to prevent the trigger from becoming invalid and missing DDL operations. If you remove the marker table, the following error is generated:

```
ORA-04098: trigger 'SYS.GGS_DDL_TRIGGER_BEFORE' is invalid and failed re-validation
```

The proper way to remove an Oracle GoldenGate DDL object depends on your plans for the rest of the DDL environment. To choose the correct procedure, see one of the following:

- Restoring an Existing DDL Environment to a Clean State (page 16-5)
- Removing the DDL Objects from the System (page 16-7)

16.5 Maintaining the DDL History Table

You can purge the DDL history table to control its size, but do so carefully. The DDL history table maintains the integrity of the DDL synchronization environment. Purges to this table cannot be recovered through the Oracle GoldenGate interface.

1. To prevent any possibility of DDL history loss, make regular full backups of the history table.
2. To ensure that purged DDL can be recovered, enable Oracle Flashback for the history table. Set the flashback retention time well past the point where it could be needed. For example, if your full backups are at most one week old, retain two weeks of flashback. Oracle GoldenGate can be positioned backward into the flashback for reprocessing.
3. If possible, purge the DDL history table manually to ensure that essential rows are not purged accidentally. If you require an automated purging mechanism, use the `PURGEDDLHISTORY` parameter in the Manager parameter file. You can specify maximum and minimum lengths of time to keep a row. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.

> **Note:**
>
> Temporary tables created by Oracle GoldenGate to increase performance might be purged at the same time as the DDL history table, according to the same rules. The names of these tables are derived from the name of the history table, and their purging is reported in the Manager report file. This is normal behavior.

16.6 Deleting the DDL History Table

Do not delete the DDL history table unless you want to discontinue synchronizing DDL. The history table contains a record of DDL operations that were issued. Once an Extract switches from using the DDL trigger to not using the trigger, as when source database redo compatibility is advanced to 11.2.0.4 or greater, these objects can be deleted though *not immediately*. It is imperative that all mining of the redo generated before the compatibility change be complete and that this redo not need to be mined again.

The history table and the DDL trigger are interdependent. An attempt to drop the history table fails if the DDL trigger is enabled. This is a safety measure to prevent the trigger from becoming invalid and missing DDL operations. If you remove the history table, the following error is generated:

```
ORA-04098: trigger 'SYS.GGS_DDL_TRIGGER_BEFORE' is invalid and failed re-validation
```

The proper way to remove an Oracle GoldenGate DDL object depends on your plans for the rest of the DDL environment. To choose the correct procedure, see one of the following:

- Restoring an Existing DDL Environment to a Clean State (page 16-5)
- Removing the DDL Objects from the System (page 16-7)

16.7 Purging the DDL Trace File

To prevent the DDL trace file from consuming excessive disk space, run the `ddl_cleartrace` script on a regular basis. This script deletes the trace file, but Oracle GoldenGate will create it again.

The default name of the DDL trace file is `ggs_ddl_trace.log`. It is in the `USER_DUMP_DEST` directory of Oracle. The `ddl_cleartrace` script is in the Oracle GoldenGate directory.

16.8 Applying Database Patches and Upgrades when DDL Support is Enabled

Database patches and upgrades usually invalidate the Oracle GoldenGate DDL trigger and other Oracle GoldenGate DDL objects. Before applying a database patch, do the following.

1. Log in to SQL*Plus as a user that has `SYSDBA` privileges.
2. Disable the Oracle GoldenGate DDL trigger by running the `ddl_disable` script in SQL*Plus.
3. Apply the patch.
4. Enable the DDL trigger by running the `ddl_enable` script in SQL*Plus.

> **Note:**
>
> Database upgrades and patches generally operate on Oracle objects. Because Oracle GoldenGate filters out those objects automatically, DDL from those procedures is not replicated when replication starts again.

To avoid recompile errors after the patch or upgrade, which are caused if the trigger is not disabled before the procedure, consider adding calls to `@ddl_disable` and `@ddl_enable` at the appropriate locations within your scripts.

16.9 Applying Oracle GoldenGate Patches and Upgrades when DDL support is Enabled

This section explains how to apply Oracle GoldenGate patches and upgrades when DDL support is enabled.

> **Note:**
>
> If the release notes or upgrade documentation for your Oracle GoldenGate release contain instructions similar to those provided in this section, follow those instructions instead the ones in this section. Do not use this procedure for an upgrade from an Oracle GoldenGate version that does not support DDL statements that are larger than 30K (pre-version 10.4). To upgrade in that case, follow the instructions in Restoring an Existing DDL Environment to a Clean State (page 16-5).

Use the following steps to apply a patch or upgrade to the DDL objects. This procedure may or may not preserve the current DDL synchronization configuration, depending on whether the new build requires a clean installation.

1. Run GGSCI. Keep the session open for the duration of this procedure.
2. Stop Extract to stop DDL capture.

   ```
   STOP EXTRACT group
   ```
3. Stop Replicat to stop DDL replication.

   ```
   STOP REPLICAT group
   ```
4. Download or extract the patch or upgrade files according to the instructions provided by Oracle GoldenGate.
5. Change directories to the Oracle GoldenGate installation directory.
6. Log in to SQL*Plus as a user that has `SYSDBA` privileges.
7. Disconnect all sessions that ever issued DDL, including those of Oracle GoldenGate processes, SQL*Plus, business applications, and any other software that uses Oracle. Otherwise the database might generate an ORA-04021 error.
8. Run the `ddl_disable` script to disable the DDL trigger.
9. Run the `ddl_setup` script. You are prompted for the name of the Oracle GoldenGate DDL schema. If you changed the schema name, use the new one.
10. Run the `ddl_enable.sql` script to enable the DDL trigger.
11. In GGSCI, start Extract to resume DDL capture.

    ```
    START EXTRACT group
    ```
12. Start Replicat to start DDL replication.

    ```
    START REPLICAT group
    ```

16.10 Restoring an Existing DDL Environment to a Clean State

Follow these steps to completely remove, and then reinstall, the Oracle GoldenGate DDL objects. This procedure creates a new DDL environment and removes any current DDL history.

> **Note:**
>
> Due to object interdependencies, all objects must be removed and reinstalled in this procedure.

1. If you are performing this procedure in conjunction with the installation of a new Oracle GoldenGate version, download and install the Oracle GoldenGate files, and create or update process groups and parameter files as necessary.
2. (Optional) To preserve the continuity of source and target structures, stop DDL activities and then make certain that Replicat finished processing all of the DDL and DML data in the trail. To determine when Replicat is finished, issue the following command until you see a message that there is no more data to process.

   ```
   INFO REPLICAT group
   ```

   > **Note:**
   >
   > Instead of using `INFO REPLICAT`, you can use the `EVENTACTIONS` option of `TABLE` and `MAP` to stop the Extract and Replicat processes after the DDL and DML has been processed.

3. Run GGSCI.
4. Stop Extract to stop DDL capture.

   ```
   STOP EXTRACT group
   ```

5. Stop Replicat to stop DDL replication.

   ```
   STOP REPLICAT group
   ```

6. Change directories to the Oracle GoldenGate installation directory.
7. Log in to SQL*Plus as a user that has `SYSDBA` privileges.
8. Disconnect all sessions that ever issued DDL, including those of Oracle GoldenGate processes, SQL*Plus, business applications, and any other software that uses Oracle. Otherwise the database might generate an ORA-04021 error.
9. Run the `ddl_disable` script to disable the DDL trigger.
10. Run the `ddl_remove` script to remove the Oracle GoldenGate DDL trigger, the DDL history and marker tables, and other associated objects. This script produces a `ddl_remove_spool.txt` file that logs the script output and a `ddl_remove_set.txt` file that logs environment settings in case they are needed for debugging.
11. Run the `marker_remove` script to remove the Oracle GoldenGate marker support system. This script produces a `marker_remove_spool.txt` file that logs the script output and a `marker_remove_set.txt` file that logs environment settings in case they are needed for debugging.
12. If you are changing the DDL schema for this installation, grant the following permission to the Oracle GoldenGate schema.

    ```
    GRANT EXECUTE ON utl_file TO schema;
    ```

13. If you are changing the DDL schema for this installation, the schema's default tablespace must be dedicated to that schema; do not allow any other schema to share it. `AUTOEXTEND` must be set to `ON` for this tablespace, and the tablespace must be sized to accommodate the growth of the `GGS_DDL_HIST` and `GGS_MARKER` tables. The `GGS_DDL_HIST` table, in particular, will grow in proportion to overall DDL activity.

    > **Note:**
    >
    > If the DDL tablespace fills up, Extract stops capturing DDL. To cause user DDL activity to fail when that happens, edit the `params.sql` script and set the `ddl_fire_error_in_trigger` parameter to `TRUE`. Stopping user DDL gives you time to extend the tablespace size and prevent the loss of DDL capture. Managing tablespace sizing this way, however, requires frequent monitoring of the business applications and Extract to avoid business disruptions. Instead, Oracle recommends that you size the tablespace appropriately and set `AUTOEXTEND` to `ON` so that the tablespace does not fill up.

    > **WARNING:**
    >
    > Do not edit any other parameters in `params.sql` except if you need to follow documented instructions to change certain object names.

14. If you are changing the DDL schema for this installation, edit the `GLOBALS` file and specify the new schema name with the following parameter.

    ```
    GGSCHEMA schema_name
    ```

15. Run the `marker_setup` script to reinstall the Oracle GoldenGate marker support system. You are prompted for the name of the Oracle GoldenGate schema.
16. Run the `ddl_setup` script. You are prompted for the name of the Oracle GoldenGate DDL schema.
17. Run the `role_setup` script to recreate the Oracle GoldenGate DDL role.
18. Grant the role to all Oracle GoldenGate users under which the following Oracle GoldenGate processes run: Extract, Replicat, GGSCI, and Manager. You might need to make multiple grants if the processes have different user names.
19. Run the `ddl_enable.sql` script to enable the DDL trigger.

16.11 Removing the DDL Objects from the System

This procedure removes the DDL environment and removes the history that maintains continuity between source and target DDL operations.

> **Note:**
>
> Due to object interdependencies, all objects must be removed.

1. Run GGSCI.
2. Stop Extract to stop DDL capture.

   ```
   STOP EXTRACT group
   ```

3. Stop Replicat to stop DDL replication.

   ```
   STOP REPLICAT group
   ```

4. Change directories to the Oracle GoldenGate installation directory.
5. Run SQL*Plus and log in as a user that has `SYSDBA` privileges.
6. Disconnect all sessions that ever issued DDL, including those of Oracle GoldenGate processes, SQL*Plus, business applications, and any other software that uses Oracle. Otherwise the database might generate an ORA-04021 error.
7. Run the `ddl_disable` script to disable the DDL trigger.
8. Run the `ddl_remove` script to remove the Oracle GoldenGate DDL trigger, the DDL history and marker tables, and the associated objects. This script produces a `ddl_remove_spool.txt` file that logs the script output and a `ddl_remove_set.txt` file that logs current user environment settings in case they are needed for debugging.
9. Run the `marker_remove` script to remove the Oracle GoldenGate marker support system. This script produces a `marker_remove_spool.txt` file that logs the script output and a `marker_remove_set.txt` file that logs environment settings in case they are needed for debugging.

17
Uninstalling Oracle GoldenGate

This chapter contains instructions for removing Oracle GoldenGate processes and files from the local system.
This chapter includes the following sections:

This procedure assumes that you no longer need the data in the Oracle GoldenGate trails, and that you no longer need to preserve the current Oracle GoldenGate environment. To preserve your current environment and data, make a backup of the Oracle GoldenGate directory and all subdirectories before starting this procedure.

Topics:

17.1 Stopping Processes

This procedure stops the Extract and Replication processes. Leave Manager running until directed to stop it.

On all Systems:

1. Run the command shell.
2. Log on as the system administrator or as a user with permission to issue Oracle GoldenGate commands and delete files and directories from the operating system.
3. Change directories to the Oracle GoldenGate installation directory.
4. Run GGSCI.
5. Stop all Oracle GoldenGate processes.

   ```
   STOP ER *
   ```

6. Stop the Manager process.

   ```
   STOP MANAGER
   ```

17.2 Removing the DDL Environment

(Valid when the DDL trigger is being used to support DDL replication.) This procedure removes all of the Oracle GoldenGate DDL objects from the DDL schema on a source system.

1. Log on as the system administrator or as a user with permission to issue Oracle GoldenGate commands and delete files and directories from the operating system.

2. Run GGSCI from the Oracle GoldenGate directory.
3. Stop all Oracle GoldenGate processes.

   ```
   STOP ER *
   ```

4. Log in to SQL*Plus as a user that has `SYSDBA` privileges.
5. Disconnect all sessions that ever issued DDL, including those of Oracle GoldenGate processes, SQL*Plus, business applications, and any other software that uses Oracle. Otherwise the database might generate an ORA-04021 error.
6. Run the `ddl_disable` script to disable the DDL trigger.
7. Run the `ddl_remove` script to remove the Oracle GoldenGate DDL trigger, the DDL history and marker tables, and other associated objects. This script produces a `ddl_remove_spool.txt` file that logs the script output and a `ddl_remove_set.txt` file that logs environment settings in case they are needed for debugging.
8. Run the `marker_remove` script to remove the Oracle GoldenGate marker support system. This script produces a `marker_remove_spool.txt` file that logs the script output and a `marker_remove_set.txt` file that logs environment settings in case they are needed for debugging.

17.3 Removing Database Objects

Follow these instructions to remove Oracle GoldenGate objects that are configured within an Oracle Database. Specific steps and commands may not apply to your configuration.

On a Source System:

1. Log on as the system administrator or as a user with permission to issue Oracle GoldenGate commands and delete files and directories from the operating system.
2. Run GGSCI from the Oracle GoldenGate directory.
3. Stop all Oracle GoldenGate processes.

   ```
   STOP ER *
   ```

4. Stop the Manager process.

   ```
   STOP MANAGER
   ```

5. In GGSCI, log into the database with the `DBLOGIN` (or the `MININGDBLOGIN` command if you need to remove a database logmining server from a downstream mining database). `[MINING]DBLOGIN` requires privileges granted in the `dbms_goldengate_auth.grant_admin_privilege` procedure.

   ```
   [MINING]DBLOGIN USERIDALIAS alias
   ```

6. In GGSCI, run any or all of the following commands, depending on your configuration. See *Reference for Oracle GoldenGate for Windows and UNIX*.
   - Disable schema-level supplemental logging (wildcards not allowed):

     ```
     DELETE SCHEMATRANDATA schema [NOSCHEDULINGCOLS | ALLCOLS]
     ```

   - Disable table-level supplemental logging.

     ```
     DELETE TRANDATA [container.]schema.table  [NOSCHEDULINGCOLS | ALLCOLS]
     ```

   - (Bidirectional configuration) Remove the Oracle trace table.

```
DELETE TRACETABLE [container.]schema.table
```

- (Classic capture configuration) Disable log retention and remove the underlying Oracle Streams capture process. `DBLOGIN` requires privileges shown in Table 11-3 (page 11-11) on .

  ```
  UNREGISTER EXTRACT group LOGRETENTION
  ```

- (Integrated capture configuration) Remove the database logmining server from an Oracle mining database.

  ```
  DELETE EXTRACT group
  UNREGISTER EXTRACT group DATABASE
  ```

7. Run the following Oracle procedure to remove the privileges from the Oracle GoldenGate admin users for both classic and integrated processes.

   ```
   dbms_goldengate_auth.revoke_admin_privilege('ggadm')
   ```

On a Target System:

1. Stop Replicat.

   ```
   STOP REPLICAT group
   ```

2. Log into the database with the `DBLOGIN` command.

   ```
   DBLOGIN USERIDALIAS alias
   ```

3. (Integrated Replicat) Delete the Replicat group. This command also deletes the inbound server from the target database.

   ```
   DELETE REPLICAT group
   ```

4. (Nonintegrated Replicat) Remove the Replicat checkpoint table by running the `DELETE CHECKPOINTTABLE` command.

   ```
   DELETE CHECKPOINTTABLE [container.]schema.table
   ```

17.4 Uninstalling Oracle GoldenGate Using Oracle Universal Installer

Follow these instructions to uninstall Oracle GoldenGate through an interactive session of Oracle Universal Installer (OUI).

> **WARNING:**
>
> Before removing Oracle GoldenGate through OUI, follow the instructions in Removing the DDL Environment (page 17-1) (if using trigger-based DDL capture) and Removing Database Objects (page 17-2). These procedures require the use of Oracle GoldenGate commands and scripts, which are removed by the OUI uninstaller.

The following items are removed in this process.

- The Oracle GoldenGate home directory in the Oracle central inventory.

- The Oracle GoldenGate installation directory.
- The Oracle GoldenGate Manager service, if installed on Windows.
- The Oracle GoldenGate Windows Registry entries

To remove Oracle GoldenGate from the system:

1. Log on as the system administrator or as a user with permission to issue Oracle GoldenGate commands and delete files and directories from the operating system.
2. Run GGSCI from the Oracle GoldenGate directory.
3. Stop all Oracle GoldenGate processes.

   ```
   STOP ER *
   ```
4. Stop the Manager process.

   ```
   STOP MANAGER
   ```
5. Run the following script from the Oracle GoldenGate installation directory.

   UNIX and Linux:

   ```
   OGG_home/deinstall/deinstall.sh
   ```

   Windows:

   ```
   OGG_home/deinstall/deinstall.bat
   ```

17.5 Uninstalling Oracle GoldenGate Manually

Follow these instructions to remove the Oracle GoldenGate environment from the system manually through the operating system.

- Manually Removing Oracle GoldenGate Windows Components (page 17-4)
- Manually Removing the Oracle GoldenGate Files (page 17-5)

17.5.1 Manually Removing Oracle GoldenGate Windows Components

(Valid for Windows installations) This procedure does the following: removes Oracle GoldenGate as a Windows cluster resource from a source or target Windows system, stops Oracle GoldenGate events from being reported to the Windows Event Manager, and removes the Manager service. Perform these steps on source and target systems.

1. Log on as the system administrator or as a user with permission to issue Oracle GoldenGate commands and to delete files and directories from the operating system.
2. (Cluster) Working from the node in the cluster that owns the cluster group that contains the Manager resource, run GGSCI and make certain that all Extract and Replicat processes are stopped. Stop any that are running.

   ```
   STATUS ER *
   STOP ER *
   ```
3. (Cluster) Use the Cluster Administrator tool to take the Manager resource offline.
4. (Cluster) Right click the resource and select **Delete** to remove it.
5. Click **Start** then **Run**, and then type `cmd` in the Run dialog box to open the command console.

6. Change directories to the Oracle GoldenGate installation directory.
7. Run the `INSTALL` utility with the following syntax.

   ```
   install deleteevents deleteservice
   ```

8. (Cluster) Move the cluster group to the next node in the cluster, and repeat from 5 (page 17-4).

17.5.2 Manually Removing the Oracle GoldenGate Files

Perform these steps on all systems to remove the Oracle GoldenGate installation directory.

1. In GGSCI, verify that all processes are stopped. Stop any that are running.

   ```
   STATUS MANAGER
   STATUS ER *
   STOP MANAGER
   STOP ER *
   ```

2. Exit GGSCI.

   ```
   EXIT
   ```

3. Remove the Oracle GoldenGate installation directory.

A

Optional Parameters for Integrated Modes

This appendix contains optional parameters that may be required when operating Extract in integrated capture mode or Replicat in integrated Replicat mode.
Topics:

A.1 Additional Parameter Options for Integrated Capture

This section contains additional parameters that may be required for your Extract configuration.

Integrated capture uses a database logmining server in the mining database to mine the redo stream of the source database. You can set parameters that are specific to the logmining server by using the `TRANLOGOPTIONS` parameter with the `INTEGRATEDPARAMS` option in the Extract parameter file.

> **Note:**
>
> For detailed information and usage guidance for these parameters, see the "DBMS_CAPTURE_ADM" section in *Oracle Database PL/SQL Packages and Types Reference*.

The following parameters can be set with `INTEGRATEDPARAMS`:

- `CAPTURE_IDKEY_OBJECTS`: Controls the capture of objects that can be supported by `FETCH`. The default for Oracle GoldenGate is `Y` (capture ID key logical change records).
- `DOWNSTREAM_REAL_TIME_MINE`: Controls whether the logmining server operates as a real-time downstream capture process or as an archived-log downstream capture process. The default is `N` (archived-log mode). Specify this parameter to use real-time capture in a downstream logmining server configuration. For more information on establishing a downstream mining configuration, see Configuring a Downstream Mining Database (page B-1).
- `INLINE_LOB_OPTIMIZATION`: Controls whether LOBs that can be processed inline (such as small LOBs) are included in the LCR directly, rather than sending LOB chunk LCRs. The default for Oracle GoldenGate is `Y` (Yes).
- `MAX_SGA_SIZE`: Controls the amount of shared memory used by the logmining server. The shared memory is obtained from the streams pool of the SGA. The default is 1 GB.
- `PARALLELISM`: Controls the number of processes used by the logmining server. The default is 2. For Oracle Standard Edition, this must be set to `1`.

- `TRACE_LEVEL`: Controls the level of tracing for the Extract logmining server. For use only with guidance from Oracle Support. The default for Oracle GoldenGate is `0` (no tracing).
- `WRITE_ALERT_LOG`: Controls whether the Extract logmining server writes messages to the Oracle alert log. The default for Oracle GoldenGate is `Y` (Yes).

See Managing Server Resources (page 3-10).

A.2 Additional Parameter Options for Integrated Replicat

The default Replicat configuration as directed in Configuring Oracle GoldenGate Apply (page 9-1) should be sufficient. However, if needed, you can set the following inbound server parameters to support specific requirements. You can set these parameters by using the `DBOPTIONS` parameter with the `INTEGRATEDPARAMS` option or dynamically by issuing the `SEND REPLICAT` command with the `INTEGRATEDPARAMS` option in GGSCI.

> **Note:**
>
> For detailed information and usage guidance for these parameters, see the "DBMS_APPLY_ADM" section in *Oracle Database PL/SQL Packages and Types Reference*.
>
> See *Reference for Oracle GoldenGate for Windows and UNIX* for more information about the `DBOPTIONS` parameter.

- `COMMIT_SERIALIZATION`: Controls the order in which applied transactions are committed and has 2 modes, `DEPENDENT_TRANSACTIONS` and `FULL`. The default mode for Oracle GoldenGate is `DEPENDENT_TRANSACTIONS` where dependent transactions are applied in the correct order though may not necessarily be applied in source commit order. In `FULL` mode, the source commit order is enforced when applying transactions.
- `BATCHSQL_MODE`: Controls the batch execution scheduling mode including pending dependencies. A pending dependency is a dependency on another transaction that has already been scheduled, but not completely executed. The default is `DEPENDENT`. You can use following three modes:

  `DEPENDENT`

  Dependency aware scheduling without an early start. Batched transactions are scheduled when there are no pending dependencies.

  `DEPENDENT_EAGER`

  Dependency aware batching with early start. Batched transactions are scheduled irrespective of pending dependencies.

  `SEQUENTIAL`

  Sequential batching. Transactions are batched by grouping the transactions sequentially based on the original commit order.

- `DISABLE_ON_ERROR`: Determines whether the apply server is disabled or continues on an unresolved error. The default for Oracle GoldenGate is `N` (continue on errors). The default setting enables Replicat to perform error handling based on the

REPERROR parameter or the Conflict Detection and Resolution (CDR) parameters, if used.

- EAGER_SIZE: Sets a threshold for the size of a transaction (in number of LCRs) after which Oracle GoldenGate starts applying data before the commit record is received. The default for Oracle GoldenGate is 15100.
- ENABLE_XSTREAM_TABLE_STATS: Controls whether statistics on applied transactions are recorded in the V$GOLDENGATE_TABLE_STATS view or not collected at all. The default for Oracle GoldenGate is Y (collect statistics).
- MAX_PARALLELISM: Limits the number of apply servers that can be used when the load is heavy. This number is reduced again when the workload subsides. The automatic tuning of the number of apply servers is effective only if PARALLELISM is greater than 1 and MAX_PARALLELISM is greater than PARALLELISM. If PARALLELISM is equal to MAX_PARALLELISM, the number of apply servers remains constant during the workload. The default for Oracle GoldenGate is 50.
- MAX_SGA_SIZE: Controls the amount of shared memory used by the inbound server. The shared memory is obtained from the streams pool of the SGA. The default for Oracle GoldenGate is INFINITE.
- MESSAGE_TRACKING_FREQUENCY: Controls how often LCRs are marked for high-level LCR tracing through the apply processing. The default value is 2000000, meaning that every 2 millionth LCR is traced. A value of zero (0) disables LCR tracing.
- PARALLELISM: Sets a minimum number of apply servers that can be used under normal conditions. Setting PARALLELISM to 1 disables apply parallelism, and transactions are applied with a single apply server process. The default for Oracle GoldenGate is 4. For Oracle Standard Edition, this must be set to 1.
- PARALLELISM_INTERVAL: Sets the interval in seconds at which the current workload activity is computed. Replicat calculates the mean throughput every 5 X PARALLELISM_INTERVAL seconds. After each calculation, the apply component can increase or decrease the number of apply servers to try to improve throughput. If throughput is improved, the apply component keeps the new number of apply servers. The parallelism interval is used only if PARALLELISM is set to a value greater than one and the MAX_PARALLELISM value is greater than the PARALLELISM value. The default is 5 seconds.
- PRESERVE_ENCRYPTION: Controls whether to preserve encryption for columns encrypted using Transparent Data Encryption. The default for Oracle GoldenGate is N (do not apply the data in encrypted form).
- OPTIMIZE_PROGRESS_TABLE: Integrated Delivery uses this table to track the transactions that have been applied. It is used for duplicate avoidance in the event of failure or restart. If it is set to N (the default), then the progress table is updated synchronously with the apply of each replicated transaction. When set to Y, rather than populating the progress table synchronously, markers are dropped into the redo stream so when the apply process starts up, it mines the redo logs for these markers, and then updates the progress table for the previously applied transactions.
- TRACE_LEVEL: Controls the level of tracing for the Replicat inbound server. For use only with guidance from Oracle Support. The default for Oracle GoldenGate is 0 (no tracing).
- WRITE_ALERT_LOG: Controls whether the Replicat inbound server writes messages to the Oracle alert log. The default for Oracle GoldenGate is Y (yes).

B

Configuring a Downstream Mining Database

This appendix contains instructions for preparing a downstream Oracle mining database to support Extract in integrated capture mode.
For more information about integrated capture, see Deciding Which Capture Method to Use (page 5-2).

For examples of the downstream mining configuration, see Example Downstream Mining Configuration (page C-1).

Topics:

B.1 Evaluating Capture Options for a Downstream Deployment

Downstream deployment allows you to offload the source database. The source database ships its redo logs to a downstream database, and Extract uses the logmining server at the downstream database to mine the redo logs. A downstream mining database can accept both archived logs and online redo logs from a source database.

Multiple source databases can send their redo data to a single downstream database; however the downstream mining database can accept *online* redo logs from only one of those source databases. The rest of the source databases must ship archived logs.

When online logs are shipped to the downstream database, *real-time capture* by Extract is possible. Changes are captured as though Extract is reading from the source logs. In order to accept online redo logs from a source database, the downstream mining database must have standby redo logs configured.

When using a downstream mining configuration, the source database and mining database must be of the same platform. For example, if the source database is running on Linux 64-bit, the downstream database must also be on the Linux 64-bit platform.

B.2 Preparing the Source Database for Downstream Deployment

This section guides you in the process of:

B.2.1 Creating the Source User Account

There must be an Extract user on the source database. Extract uses the credentials of this user to do metadata queries and to fetch column values as needed from the source database. The source user is specified by the `USERIDALIAS` parameter.

To assign the required privileges, follow the procedure in Establishing Oracle GoldenGate Credentials (page 4-1)

B.2.2 Configuring Redo Transport from Source to Downstream Mining Database

Complete the following steps to set up the transfer of redo log files from a source database to the downstream mining database, and to prepare the downstream mining database to accept these redo log files.

> **Note:**
>
> The archived logs shipped from the source databases are called *foreign archived logs*. You must not use the recovery area at the downstream mining database to store foreign archived logs. Such a configuration is not supported by Integrated Capture.

These instructions take into account the requirements to ship redo from multiple sources, if required. You configure an Extract process for each of those sources. The following summarizes the rules for supporting multiple sources sending redo to a single downstream mining database:

- Only one source database can be configured to send *online* redo to the standby redo logs at the downstream mining database. The `log_archive_dest_n` setting for this source database should *not* have a `TEMPLATE` clause.
- Source databases that are *not* sending online redo to the standby redo logs of the downstream mining database *must have* a `TEMPLATE` clause specified in the `log_archive_dest_n` parameter.
- Each of the source databases that sends redo to the downstream mining database must have a unique DBID. You can select the `DBID` column from the `v$database` view of these source databases to ensure that the DBIDs are unique.
- The `FAL_SERVER` value must be set to the downstream mining database. `FAL_SERVER` specifies the FAL (fetch archive log) server for a standby database. The value is a list of Oracle Net service names, which are assumed to be configured properly on the standby database system to point to the desired FAL servers. The list contains the net service name of any database that can potentially ship redo to the downstream database.

To Configure Redo Transport

1. Configure Oracle Net so that each source database can communicate with the mining database. For instructions, see *Oracle Database Net Services Administrator's Guide*.
2. Configure authentication at each source database and at the downstream mining database to support the transfer of redo data. Redo transport sessions are authenticated using either the Secure Sockets Layer (SSL) protocol or a remote login password file. If a source database has a remote login password file, copy it to the appropriate directory of the mining database system. The password file must be the same at all source databases, and at the mining database. For more information about authentication requirements for redo transport, see "Preparing the Primary Database for Standby Database Creation" in *Oracle Data Guard Concepts and Administration*.
3. At each source database, configure one `LOG_ARCHIVE_DEST_n` initialization parameter to transmit redo data to the downstream mining database. Set the attributes of this parameter as shown in one of the following examples, depending on whether real-time or archived-log-only capture mode is to be used.
   - Example for real-time capture at the downstream logmining server, where the source database sends its online redo logs to the downstream database:

     ```
     ALTER SYSTEM
     SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC NOREGISTER
     VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)DB_UNIQUE_NAME=dbmscap'
     ```

   - Example for archived-log-only capture at the downstream logmining server:

     ```
     ALTER SYSTEM SET
     LOG_ARCHIVE_DEST_2='SERVICE=DMBSCAP.EXAMPLE.COM ASYNC NOREGISTER
     VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
     TEMPLATE=/usr/oracle/log_for_dbms1/dbms1_arch_%t_%s_%r.log
     DB_UNIQUE_NAME=dbmscap'
     ```

   > **Note:**
   >
   > When using an archived-log-only downstream mining database, you must specify a value for the `TEMPLATE` attribute. Oracle also recommends that you use the `TEMPLATE` clause in the source databases so that the log files from all remote source databases are kept separated from the local database log files, and from each other.

4. At the source database, set a value of `ENABLE` for the `LOG_ARCHIVE_DEST_STATE_n` initialization parameter that corresponds with the `LOG_ARCHIVE_DEST_n` parameter that corresponds to the destination for the downstream mining database, as shown in the following example.

   ```
   ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE
   ```

5. At the source database, and at the downstream mining database, set the `DG_CONFIG` attribute of the `LOG_ARCHIVE_CONFIG` initialization parameter to include the `DB_UNIQUE_NAME` of the source database and the downstream database, as shown in the following example.

   ```
   ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1,dbmscap)'
   ```

B.3 Preparing the Downstream Mining Database

The following sections explain how to prepare the downstream mining database:

- Creating the Downstream Mining User Account (page B-4)
- Configuring the Mining Database to Archive Local Redo Log Files (page B-4)
- Preparing a Downstream Mining Database for Real-time Capture (page B-5)

B.3.1 Creating the Downstream Mining User Account

When using a downstream mining configuration, there must be an Extract mining user on the downstream database. The mining Extract process uses the credentials of this user to interact with the downstream logmining server. The downstream mining user is specified by the `TRANLOGOPTIONS` parameter with the `MININGUSERALIAS` option. See Establishing Oracle GoldenGate Credentials (page 4-1) to assign the correct credentials for the version of your database.

B.3.2 Configuring the Mining Database to Archive Local Redo Log Files

This procedure configures the downstream mining database to archive redo data in its online redo logs. These are redo logs that are generated at the downstream mining database.

Archiving must be enabled at the downstream mining database if you want to run Extract in real-time integrated capture mode, but it is also recommended for archive-log-only capture. Extract in integrated capture mode writes state information in the database. Archiving and regular backups will enable you to recover this state information in case there are disk failures or corruption at the downstream mining database.

To Archive Local Redo Log Files

1. Alter the downstream mining database to be in archive log mode. You can do this by issuing the following DDL.

   ```
   STARTUP MOUNT;
   ALTER DATABASE ARCHIVELOG;
   ALTER DATABASE OPEN;
   ```

2. At the downstream mining database, set the first archive log destination in the `LOG_ARCHIVE_DEST_n` initialization parameter as shown in the following example:

   ```
   ALTER SYSTEM SET
   LOG_ARCHIVE_DEST_1='LOCATION=/home/arc_dest/local
   VALID_FOR=(ONLINE_LOGFILE,PRIMARY_ROLE)'
   ```

   Alternatively, you can use a command like this example:

```
ALTER SYSTEM SET
LOG_ARCHIVE_DEST_1='LOCATION='USE_DB_RECOVERY_FILE_DEST'
valid_for=(ONLINE_LOGFILE,PRIMARY_ROLE)'
```

> **Note:**
>
> The online redo logs generated by the downstream mining database can be archived to a recovery area. However, you must not use the recovery area of the downstream mining database to stage foreign archived logs or to archive standby redo logs. For information about configuring a fast recovery area, see *Oracle Database Backup and Recovery User's Guide*.

3. Enable the local archive destination.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1=ENABLE
```

For more information about these initialization parameters, see *Oracle Data Guard Concepts and Administration*.

B.3.3 Preparing a Downstream Mining Database for Real-time Capture

This procedure is only required if you want to use real-time capture at a downstream mining database. It is not required to use archived-log-only capture mode. To use real-time capture, it is assumed that the downstream database has already been configured to archive its local redo data as shown in Configuring the Mining Database to Archive Local Redo Log Files (page B-4).

- Create the Standby Redo Log Files (page B-5)
- Configure the Database to Archive Standby Redo Log Files Locally (page B-7)

B.3.3.1 Create the Standby Redo Log Files

The following steps outline the procedure for adding standby redo log files to the downstream mining database. The following summarizes the rules for creating the standby redo logs:

- Each standby redo log file must be at least as large as the largest redo log file of the redo source database. For administrative ease, Oracle recommends that all redo log files at source database and the standby redo log files at the downstream mining database be of the same size.
- The standby redo log must have at least one more redo log group than the redo log at the source database, for each redo thread at the source database.

The specific steps and SQL statements that are required to add standby redo log files depend on your environment. See Oracle Data Guard Concepts and Administration 11g Release 2 (11.2) for detailed instructions about adding standby redo log files to a database.

> **Note:**
>
> If there will be multiple source databases sending redo to a single downstream mining database, only one of those sources can send redo to the standby redo logs of the mining database. An Extract process that mines the redo from this source database can run in real-time mode. All other source databases must send only their archived logs to the downstream mining database, and the Extracts that read this data must be configured to run in archived-log-only mode.

To Create the Standby Redo Log Files

1. In SQL*Plus, connect to the source database as an administrative user.
2. Determine the size of the source log file. Make note of the results.

   ```
   SELECT BYTES FROM V$LOG;
   ```

3. Determine the number of online log file groups that are configured on the source database. Make note of the results.

   ```
   SELECT COUNT(GROUP#) FROM V$LOG;
   ```

4. Connect to the downstream mining database as an administrative user.
5. Add the standby log file groups to the mining database. The standby log file size must be at least the size of the source log file size. The number of standby log file groups must be at least one more than the number of source online log file groups. This applies to each instance (thread) in a RAC installation. So if you have "n" threads at the source database, each having "m" redo log groups, you should configure n*(m+1) redo log groups at the downstream mining database.

   The following example shows three standby log groups.

   ```
   ALTER DATABASE ADD STANDBY LOGFILE GROUP 3
   ('/oracle/dbs/slog3a.rdo', '/oracle/dbs/slog3b.rdo') SIZE 500M;
   ALTER DATABASE ADD STANDBY LOGFILE GROUP 4
   ('/oracle/dbs/slog4.rdo', '/oracle/dbs/slog4b.rdo') SIZE 500M;
   ALTER DATABASE ADD STANDBY LOGFILE GROUP 5
   ('/oracle/dbs/slog5.rdo', '/oracle/dbs/slog5b.rdo') SIZE 500M;
   ```

6. Confirm that the standby log file groups were added successfully.

   ```
   SELECT GROUP#, THREAD#, SEQUENCE#, ARCHIVED, STATUS
   FROM V$STANDBY_LOG;
   ```

   The output should be similar to the following:

   ```
   GROUP#     THREAD#    SEQUENCE#  ARC STATUS
   ---------- ---------- ---------- --- ----------
            3          0          0 YES UNASSIGNED
            4          0          0 YES UNASSIGNED
            5          0          0 YES UNASSIGNED
   ```

7. Ensure that log files from the source database are appearing in the location that is specified in the `LOCATION` attribute of the local `LOG_ARCHIVE_DEST_n` that you set. You might need to switch the log file at the source database to see files in the directory.

B.3.3.2 Configure the Database to Archive Standby Redo Log Files Locally

This procedure configures the downstream mining database to archive the standby redo logs that receive redo data from the online redo logs of the source database. Keep in mind that foreign archived logs should not be archived in the recovery area of the downstream mining database.

To Archive Standby Redo Logs Locally

1. At the downstream mining database, set the second archive log destination in the `LOG_ARCHIVE_DEST_n` initialization parameter as shown in the following example.

   ```
   ALTER SYSTEM SET
   LOG_ARCHIVE_DEST_2='LOCATION=/home/arc_dest/srl_dbms1
   VALID_FOR=(STANDBY_LOGFILE,PRIMARY_ROLE)'
   ```

   Oracle recommends that foreign archived logs (logs from remote source databases) be kept separate from local mining database log files, and from each other. You must not use the recovery area of the downstream mining database to stage foreign archived logs. For information about configuring a fast recovery area, see *Oracle Database Backup and Recovery User's Guide*.

2. Enable the `LOG_ARCHIVE_DEST_2` parameter you set in the previous step as shown in the following example.

   ```
   ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE
   ```

   For more information about these initialization parameters, see *Oracle Data Guard Concepts and Administration*.

C

Example Downstream Mining Configuration

This appendix contains examples for preparing a downstream Oracle mining database to support Extract in integrated capture mode.
Configuring a downstream mining database, see Configuring a Downstream Mining Database (page B-1).

Topics:

C.1 Example 1: Capturing from One Source Database in Real-time Mode

> **Note:**
>
> This example assumes that you created the necessary standby redo log files as shown in Configuring a Downstream Mining Database (page B-1).

This example captures changes from source database DBMS1 by deploying an integrated capture session at a downstream mining database DBMSCAP. This assumes that the following users exist:

- User GGADM1 in DBMS1 whose credentials Extract will use to fetch data and metadata from DBMS1. This user has the alias of `ggadm1` in the Oracle GoldenGate credential store and logs in as `ggadm1@dbms1`. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at the source database.
- User GGADMCAP in DBMSCAP whose credentials Extract will use to retrieve logical change records from the logmining server at the downstream mining database DBMSCAP. This user has the alias of `ggadmcap` in the Oracle GoldenGate credential store and logs in as `ggadmcap@dbmscap`. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at the mining database.
- Prepare the Mining Database to Archive its Local Redo (page C-2)
- Prepare the Mining Database to Archive Redo Received in Standby Redo Logs from the Source Database (page C-2)

- Prepare the Source Database to Send Redo to the Mining Database (page C-2)
- Set up Integrated Capture (ext1) on DBMSCAP (page C-3)

C.1.1 Prepare the Mining Database to Archive its Local Redo

To prepare the mining database to archive its local redo:

1. The downstream mining database must be in archive log mode. You can do this by issuing the following DDL.

```
STARTUP MOUNT;
ALTER DATABASE ARCHIVELOG;
ALTER DATABASE OPEN;
```

2. At the downstream mining database, set `log_archive_dest_1` to archive local redo.

```
ALTER SYSTEM SETLOG_ARCHIVE_DEST_1='LOCATION=/home/arc_dest/
localVALID_FOR=(ONLINE_LOGFILE, PRIMARY_ROLE)'
```

3. Enable `log_archive_dest_1`.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1=ENABLE
```

C.1.2 Prepare the Mining Database to Archive Redo Received in Standby Redo Logs from the Source Database

To prepare the mining database to archive the redo received in standby redo logs from the source database:

1. At the downstream mining database, set `log_archive_dest_2` as shown in the following example.

```
ALTER SYSTEM SET
LOG_ARCHIVE_DEST_2='LOCATION=/home/arc_dest/srl_dbms1
VALID_FOR=(STANDBY_LOGFILE,PRIMARY_ROLE)'
```

2. Enable `log_archive_dest_2` as shown in the following example.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

3. Set `DG_CONFIG` at the downstream mining database.

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1,dbmscap)'
```

C.1.3 Prepare the Source Database to Send Redo to the Mining Database

To prepare the source database to send redo to the mining database:

1. Make sure that the source database is running with the required compatibility.

```
select name, value from v$parameter where name = 'compatible';

NAME            VALUE
---------       ---------------------
compatible      11.1.0.7.0
```

   The minimum compatibility setting required from integrated capture is 11.1.0.0.0.

2. Set `DG_CONFIG` at the source database.

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1,dbmscap)';
```

3. Set up redo transport at the source database.

```
ALTER SYSTEM
SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC OPTIONAL NOREGISTER
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)DB_UNIQUE_NAME=dbmscap';
```

4. Enable the downstream destination.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

C.1.4 Set up Integrated Capture (ext1) on DBMSCAP

To set up integrated capture (ext1) on DBMSCAP:

1. Register Extract with the downstream mining database. In the credential store, the alias name of `ggadm1` is linked to a user connect string of `ggadm1@dbms1`. The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`.

```
GGSCI> DBLOGIN USERIDALIAS ggadm1
GGSCI> MININGDBLOGIN USERIDALIAS ggadmcap
GGSCI> REGISTER EXTRACT ext1 DATABASE
```

2. Create Extract at the downstream mining database.

```
GGSCI> ADD EXTRACT ext1 INTEGRATED TRANLOG BEGIN NOW
```

3. Edit Extract parameter file `ext1.prm`. The following lines must be present to take advantage of real-time capture. In the credential store, the alias name of `ggadm1` is linked to a user connect string of `ggadm1@dbms1`. The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`.

```
USERIDALIAS ggadm1
TRANLOGOPTIONS MININGUSERALIAS ggadmcap
TRANLOGOPTIONS INTEGRATEDPARAMS (downstream_real_time_mine Y)
```

4. Start Extract.

```
GGSCI> START EXTRACT ext1
```

> **Note:**
>
> You can create multiple Extracts running in real-time integrated capture mode in the downstream mining database, as long as they all are capturing data from the same source database, such as capturing changes for database DBMS1 in the preceding example.

C.2 Example 2: Capturing from Multiple Sources in Archive-log-only Mode

The following example captures changes from database DBMS1 and DBMS2 by deploying an integrated capture session at a downstream mining database DBMSCAP. It assumes the following users:

- User GGADM1 in DBMS1 whose credentials Extract will use to fetch data and metadata from DBMS1. It is assumed that the

`DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at DBMS1.

- User GGADM2 in DBMS2 whose credentials Extract will use to fetch data and metadata from DBMS2. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at DBMS2.
- User GGADMCAP in DBMSCAP whose credentials Extract will use to retrieve logical change records from the logmining server at the downstream mining database. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at the downstream mining database DBMSCAP.

This procedure also assumes that the downstream mining database is configured in archive log mode.

- Prepare the Mining Database to Archive its Local Redo (page C-4)
- Prepare the Mining Database to Archive Redo from the Source Database (page C-4)
- Prepare the First Source Database to Send Redo to the Mining Database (page C-5)
- Prepare the Second Source Database to Send Redo to the Mining Database (page C-5)
- Set up Extracts at the Downstream Mining Database (page C-6)

C.2.1 Prepare the Mining Database to Archive its Local Redo

To prepare the mining database to archive its local redo:

1. The downstream mining database must be in archive log mode. You can do this by issuing the following DDL.

```
STARTUP MOUNT;
ALTER DATABASE ARCHIVELOG;
ALTER DATABASE OPEN;
```

2. At the downstream mining database, set `log_archive_dest_1` to archive local redo.

```
ALTER SYSTEM SET
LOG_ARCHIVE_DEST_1='LOCATION=/home/arc_dest/local
VALID_FOR=(ONLINE_LOGFILE, PRIMARY_ROLE)'
```

3. Enable `log_archive_dest_1`.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1=ENABLE
```

C.2.2 Prepare the Mining Database to Archive Redo from the Source Database

Set `DG_CONFIG` at the downstream mining database.

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1,dbms2, dbmscap)'
```

C.2.3 Prepare the First Source Database to Send Redo to the Mining Database

To prepare the first source database to send redo to the mining database:

1. Make certain that DBMS1 source database is running with the required compatibility.

```
select name, value from v$parameter where name = 'compatible';

NAME            VALUE
---------       ---------------------
compatible      11.1.0.7.0
```

   The minimum compatibility setting required from integrated capture is 11.1.0.0.0.

2. Set `DG_CONFIG` at DBMS1 source database.

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1, dbmscap)';
```

3. Set up redo transport at DBMS1 source database. The `TEMPLATE` clause is mandatory if you want to send redo data directly to foreign archived logs at the downstream mining database.

```
ALTER SYSTEM
SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC OPTIONAL NOREGISTER
TEMPLATE='/usr/orcl/arc_dest/dbms1/dbms1_arch_%t_%s_%r.log
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)DB_UNIQUE_NAME=dbmscap';
```

4. Enable the downstream destination.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

C.2.4 Prepare the Second Source Database to Send Redo to the Mining Database

To prepare the second source database to send redo to the mining database:

1. Make sure that DBMS2 source database is running with the required compatibility.

```
select name, value from v$parameter where name = 'compatible';

NAME            VALUE
---------       ---------------------
compatible      11.1.0.0.0
```

   The minimum compatibility setting required from integrated capture is 11.1.0.0.0.

2. Set `DG_CONFIG` at DBMS2 source database.

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms2, dbmscap)';
```

3. Set up redo transport at DBMS2 source database. The `TEMPLATE` clause is mandatory if you want to send redo data directly to foreign archived logs at the downstream mining database.

```
ALTER SYSTEM
SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC OPTIONAL NOREGISTER
TEMPLATE='/usr/orcl/arc_dest/dbms2/dbms2_arch_%t_%s_%r.log
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)DB_UNIQUE_NAME=dbmscap';
```

4. Enable the downstream destination.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

C.2.5 Set up Extracts at the Downstream Mining Database

The following steps set up Extract at the downstream database to capture from the archived logs sent by DBMS1 and DBMS2.

- Set up Extract (ext1) to Capture Changes from Archived Logs Sent by DBMS1 (page C-6)
- Set up Extract (ext2) to Capture Changes from Archived Logs Sent by DBMS2 (page C-6)

C.2.5.1 Set up Extract (ext1) to Capture Changes from Archived Logs Sent by DBMS1

Perform the following steps on the DBMSCAP downstream mining database.

1. Register Extract with DBMSCAP for the DBMS1 source database. In the credential store, the alias name of `ggadm1` is linked to a user connect string of `ggadm1@dbms1`. The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`.

```
GGSCI> DBLOGIN USERIDALIAS ggadm1
GGSCI> MININGDBLOGIN USERIDALIAS ggadmcap
GGSCI> REGISTER EXTRACT ext1 DATABASE
```

2. Add Extract at the mining database DBMSCAP.

```
GGSCI> ADD EXTRACT ext1 INTEGRATED TRANLOG BEGIN NOW
```

3. Edit the Extract parameter file `ext1.prm`. In the credential store, the alias name of `ggadm1` is linked to a user connect string of `ggadm1@dbms1`. The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`.

```
USERIDALIAS ggadm1
TRANLOGOPTIONS MININGUSERALIAS ggadmcap
TRANLOGOPTIONS INTEGRATEDPARAMS (downstream_real_time_mine N)
```

4. Start Extract.

```
GGSCI> START EXTRACT ext1
```

C.2.5.2 Set up Extract (ext2) to Capture Changes from Archived Logs Sent by DBMS2

Perform the following steps on the downstream DBMSCAP mining database.

1. Register Extract with the mining database for source database DBMS2. In the credential store, the alias name of `ggadm2` is linked to a user connect string of `ggadm2@dbms2`.The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`.

```
GGSCI> DBLOGIN USERIDALIAS ggadm2
GGSCI> MININGDBLOGIN USERIDALIAS ggadmcap
GGSCI> REGISTER EXTRACT ext2 DATABASE
```

2. Create Extract at the mining database.

```
GGSCI> ADD EXTRACT ext2 INTEGRATED TRANLOG, BEGIN NOW
```

3. Edit the Extract parameter file `ext2.prm`. In the credential store, the alias name of `ggadm2` is linked to a user connect string of `ggadm2@dbms2`. The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`.

```
USERIDALIAS ggadm2
TRANLOGOPTIONS MININGUSERALIAS ggadmcap
TRANLOGOPTIONS INTEGRATEDPARAMS (downstream_real_time_mine N)
```

4. Start Extract.

```
GGSCI> START EXTRACT ext2
```

> **Note:**
>
> You can create multiple Extracts capturing data from the same source database while running in archive-log-only mode in the downstream mining database. In the case of these examples, you can create other Extracts capturing changes for database DBMS1 and DBMS2 at the downstream mining database.

C.3 Example 3: Capturing from Multiple Sources with Mixed Real-time and Archive-log-only Mode

> **Note:**
>
> This example assumes that you created the necessary standby redo log files as shown in Configuring a Downstream Mining Database (page B-1).

The following example captures changes from database DBMS1, DBMS2 and DBMS3 by deploying an integrated capture session at a downstream mining database DBMSCAP. It assumes the following users:

- User GGADM1 in DBMS1 whose credentials Extract will use to fetch data and metadata from DBMS1. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at DBMS1.
- User GGADM2 in DBMS2 whose credentials Extract will use to fetch data and metadata from DBMS2. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at DBMS2.
- User GGADM3 in DBMS3 whose credentials Extract will use to fetch data and metadata from DBMS3. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at DBMS3.
- User GGADMCAP in DBMSCAP whose credentials Extract will use to retrieve logical change records from the logmining server at the downstream mining database. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()`

procedure was called to grant appropriate privileges to this user at the downstream mining database DBMSCAP.

This procedure also assumes that the downstream mining database is configured in archive log mode.

In this example, the redo sent by DBMS3 will be mined in real time mode, whereas the redo data sent from DBMS1 and DBMS2 will be mined in archive-log-only mode.

C.3.1 Prepare the Mining Database to Archive its Local Redo

To prepare the mining database to archive its local redo:

1. The downstream mining database must be in archive log mode. You can do this by issuing the following DDL.

```
STARTUP MOUNT;
ALTER DATABASE ARCHIVELOG;
ALTER DATABASE OPEN;
```

2. At the downstream mining database, set `log_archive_dest_1` to archive local redo.

```
ALTER SYSTEM SET
LOG_ARCHIVE_DEST_1='LOCATION=/home/arc_dest/local
VALID_FOR=(ONLINE_LOGFILE, PRIMARY_ROLE)'
```

3. Enable `log_archive_dest_1`.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1=ENABLE
```

C.3.2 Prepare the Mining Database to Accept Redo from the Source Databases

Because redo data is being accepted in the standby redo logs of the downstream mining database, the appropriate number of correctly sized standby redo logs must exist. If you did not configure the standby logs, see Configuring a Downstream Mining Database (page B-1).

1. At the downstream mining database, set the second archive log destination in the `LOG_ARCHIVE_DEST_n` initialization parameter as shown in the following example. This is needed to handle archive standby redo logs.

```
ALTER SYSTEM SET
LOG_ARCHIVE_DEST_2='LOCATION=/home/arc_dest/srl_dbms3
VALID_FOR=(STANDBY_LOGFILE,PRIMARY_ROLE)'
```

2. Enable the `LOG_ARCHIVE_DEST_STATE_2` initialization parameter that corresponds with the `LOG_ARCHIVE_DEST_2` parameter as shown in the following example.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

3. Set `DG_CONFIG` at the downstream mining database to accept redo data from all of the source databases.

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1, dbms2, dbms3, dbmscap)'
```

C.3.3 Prepare the First Source Database to Send Redo to the Mining Database

To prepare the first source database to send redo to the mining database:

1. Make certain that DBMS1 source database is running with the required compatibility.

```
select name, value from v$parameter where name = 'compatible';

NAME            VALUE
---------       ---------------------
compatible      11.1.0.0.0
```

The minimum compatibility setting required from integrated capture is 11.1.0.0.0.

2. Set `DG_CONFIG` at DBMS1 source database.

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1, dbmscap)';
```

3. Set up redo transport at DBMS1 source database. The `TEMPLATE` clause is mandatory if you want to send redo data directly to foreign archived logs at the downstream mining database.

```
ALTER SYSTEM
SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC OPTIONAL NOREGISTER
TEMPLATE='/usr/orcl/arc_dest/dbms1/dbms1_arch_%t_%s_%r.log
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)DB_UNIQUE_NAME=dbmscap';
```

4. Enable the downstream destination.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

C.3.4 Prepare the Second Source Database to Send Redo to the Mining Database

To prepare the second source database to send redo to the mining database:

1. Make sure that DBMS2 source database is running with the required compatibility.

```
select name, value from v$parameter where name = 'compatible';

NAME                    VALUE
---------               ---------------------
compatible              11.1.0.0.0
```

The minimum compatibility setting required from integrated capture is 11.1.0.0.0.

2. Set `DG_CONFIG` at DBMS2 source database.

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms2, dbmscap)';
```

3. Set up redo transport at DBMS2 source database. The `TEMPLATE` clause is mandatory if you want to send redo data directly to foreign archived logs at the downstream mining database.

```
ALTER SYSTEM
SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC OPTIONAL NOREGISTER
TEMPLATE='/usr/orcl/arc_dest/dbms2/dbms2_arch_%t_%s_%r.log
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)DB_UNIQUE_NAME=dbmscap';
```

4. Enable the downstream destination.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

C.3.5 Prepare the Third Source Database to Send Redo to the Mining Database

To prepare the third source database to send redo to the mining database:

1. Make sure that DBMS3 source database is running with the required compatibility.

```
select name, value from v$parameter where name = 'compatible';

NAME                 VALUE
---------            ---------------------
compatible           11.1.0.0.0
```

   The minimum compatibility setting required from integrated capture is 11.1.0.0.0.

2. Set `DG_CONFIG` at DBMS3 source database.

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms3, dbmscap)';
```

3. Set up redo transport at DBMS3 source database. Because DBMS3 is the source that will send its online redo logs to the standby redo logs at the downstream mining database, do not specify a `TEMPLATE` clause.

```
ALTER SYSTEM
SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC OPTIONAL NOREGISTER
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)DB_UNIQUE_NAME=dbmscap';
```

4. Enable the downstream destination.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

C.3.6 Set up Extracts at Downstream Mining Database

These steps set up Extract at the downstream database to capture from the archived logs sent by DBMS1 and DBMS2.

- Set up Extract (ext1) to Capture Changes from Archived Logs Sent by DBMS1 (page C-11)
- Set up Extract (ext2) to Capture Changes from Archived Logs Sent by DBMS2 (page C-11)
- Set up Extract (ext3) to Capture Changes in Real-time Mode from Online Logs Sent by DBMS3 (page C-12)

C.3.6.1 Set up Extract (ext1) to Capture Changes from Archived Logs Sent by DBMS1

Perform the following steps on the DBMSCAP downstream mining database.

1. Register Extract with DBMSCAP for the DBMS1 source database. In the credential store, the alias name of `ggadm1` is linked to a user connect string of `ggadm1@dbms1`.The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`.

   ```
   GGSCI> DBLOGIN USERIDALIAS ggadm1
   GGSCI> MININGDBLOGIN USERIDALIAS ggadmcap
   GGSCI> REGISTER EXTRACT ext1 DATABASE
   ```

2. Add Extract at the mining database DBMSCAP.

   ```
   GGSCI> ADD EXTRACT ext1 INTEGRATED TRANLOG BEGIN NOW
   ```

3. Edit the Extract parameter file `ext1.prm`. In the credential store, the alias name of `ggadm1` is linked to a user connect string of `ggadm1@dbms1`. The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`.

   ```
   USERIDALIAS ggadm1
   TRANLOGOPTIONS MININGUSERALIAS ggadmcap
   TRANLOGOPTIONS INTEGRATEDPARAMS (downstream_real_time_mine N)
   ```

4. Start Extract.

   ```
   GGSCI> START EXTRACT ext1
   ```

C.3.6.2 Set up Extract (ext2) to Capture Changes from Archived Logs Sent by DBMS2

Perform the following steps on the DBMSCAP downstream mining database.

1. Register Extract with the mining database for source database DBMS2. In the credential store, the alias name of `ggadm2` is linked to a user connect string of `ggadm2@dbms2`.The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`.

   ```
   GGSCI> DBLOGIN USERIDALIAS ggadm2
   GGSCI> MININGDBLOGIN USERIDALIAS ggadmcap
   GGSCI> REGISTER EXTRACT ext2 DATABASE
   ```

2. Create Extract at the mining database.

   ```
   GGSCI> ADD EXTRACT ext2 INTEGRATED TRANLOG, BEGIN NOW
   ```

3. Edit the Extract parameter file `ext2.prm`. In the credential store, the alias name of `ggadm2` is linked to a user connect string of `ggadm2@dbms2`.The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`.

   ```
   USERIDALIAS ggadm2
   TRANLOGOPTIONS MININGUSERALIAS ggadmcap
   TRANLOGOPTIONS INTEGRATEDPARAMS (downstream_real_time_mine N)
   ```

4. Start Extract.

   ```
   GGSCI> START EXTRACT ext2
   ```

C.3.6.3 Set up Extract (ext3) to Capture Changes in Real-time Mode from Online Logs Sent by DBMS3

Perform the following steps on the DBMSCAP downstream mining database.

1. Register Extract with the mining database for source database DBMS3. In the credential store, the alias name of `ggadm3` is linked to a user connect string of `ggadm3@dbms3`.The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`.

   ```
   GGSCI> DBLOGIN USERID ggadm3
   GGSCI> MININGDBLOGIN USERID ggadmcap
   GGSCI> REGISTER EXTRACT ext3 DATABASE
   ```

2. Create Extract at the mining database.

   ```
   GGSCI> ADD EXTRACT ext3 INTEGRATED TRANLOG, BEGIN NOW
   ```

3. Edit the Extract parameter file `ext3.prm`. To enable real-time mining, you must specify `downstream_real_time_mine`. In the credential store, the alias name of `ggadm3` is linked to a user connect string of `ggadm3@dbms3`.The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`.

   ```
   USERIDALIAS ggadm3
   TRANLOGOPTIONS MININGUSERALIAS ggadmcap
   TRANLOGOPTIONS INTEGRATEDPARAMS (downstream_real_time_mine Y)
   ```

4. Start Extract.

   ```
   GGSCI> START EXTRACT ext3
   ```

> **Note:**
>
> You can create multiple Extracts running in real-time integrated capture mode in the downstream mining database, as long as they all are capturing data from the same source database, such as all capturing for database DBMS3 in the preceding example.

D

Installing Trigger-Based DDL Capture

This appendix contains instructions for installing the objects that support the trigger-based method of Oracle GoldenGate DDL support.
To configure Oracle GoldenGate to capture and replicate DDL, see Configuring DDL Support (page 13-1).

> **Note:**
>
> DDL support for sequences (`CREATE`, `ALTER`, `DROP`, `RENAME`) is compatible with, but not required for, replicating the sequence values themselves. To replicate just sequence values, you do not need to install the Oracle GoldenGate DDL support environment. You can just use the `SEQUENCE` parameter in the Extract configuration.
>
> **Topics:**

D.1 When to Use Trigger-based DDL Capture

You *must* use trigger-based DDL capture when Extract will operate in the following configurations:

- Extract operates in classic capture mode against any version of Oracle Database.
- Extract operates in integrated mode against an Oracle Database version 11.2.0.3 or earlier.

If Extract will run in integrated mode against a version 11.2.0.4 or later Oracle Database, the DDL trigger is not required. By default, DDL capture is handled transparently through the database logmining server.

If Extract will capture from a multitenant container database, integrated capture mode must be used with the native DDL capture method.

See Choosing Capture and Apply Modes (page 5-1) for more information about capture modes.

See Configuring DDL Support (page 13-1) for more information about configuring DDL support.

D.2 Overview of the Objects that Support Trigger-based DDL Capture

To install the Oracle GoldenGate trigger-based DDL environment, you will be installing the database objects shown in Table D-1 (page D-2).

Table D-1 DDL Synchronization Objects

Object	Purpose	Default name
DDL marker table	Stores DDL information. This table only receives inserts.	`GGS_MARKER`
Sequence on marker table	Used for a column in the marker table.	`GGS_DDL_SEQ`
DDL history table	Stores object metadata history. This table receives inserts, updates, deletes.	`GGS_DDL_HIST`
Object ID history table	Contains object IDs of configured objects.	`GGS_DDL_HIST_ALT`
DDL trigger	Fires on DDL operations. Writes information about the operation to the marker and history tables. Installed with the trigger are some packages.	`GGS_DDL_TRIGGER_BEFORE`
DDL schema	Contains the DDL synchronization objects.	None; must be specified during installation and in the `GLOBALS` file.
User role	Establishes the role needed to execute DDL operations.	`GGS_GGSUSER_ROLE`
Internal setup table	Database table for internal use only.	`GGS_SETUP`
`ddl_pin`	Pins DDL tracing, the DDL package, and the DDL trigger for performance improvements.	`ddl_pin`
`ddl_cleartrace.sql`	Removes the DDL trace file.	`ddl_cleartrace.sql`
`ddl_status.sql`	Verifies that the Oracle GoldenGate DDL objects are installed	`ddl_status.sql`
`marker_status.sql`	Verifies that the marker table is installed.	`marker_status.sql`
`ddl_tracelevel.sql`	Sets the level for DDL tracing.	`ddl_tracelevel.sql`

D.3 Installing the DDL Objects

Follow these steps to install the database objects that support Oracle GoldenGate DDL capture.

> **Note:**
>
> When using Extract in classic mode to capture in an Active Data Guard environment, the DDL objects must be installed on the source database, not the standby.

1. Choose a schema that can contain the Oracle GoldenGate DDL objects. This schema cannot be case-sensitive.
2. Grant the following permission to the Oracle GoldenGate schema.

   ```
   GRANT EXECUTE ON utl_file TO schema;
   ```

3. Create a default tablespace for the Oracle GoldenGate DDL schema. This tablespace must be dedicated to the DDL schema; do not allow any other schema to share it.
4. Set `AUTOEXTEND` to `ON` for the DDL tablespace, and size it to accommodate the growth of the `GGS_DDL_HIST` and `GGS_MARKER` tables. The `GGS_DDL_HIST` table, in particular, will grow in proportion to overall DDL activity.
5. (Optional) To cause user DDL activity to fail when the DDL tablespace fills up, edit the `params.sql` script and set the `ddl_fire_error_in_trigger` parameter to `TRUE`. Extract cannot capture DDL if the tablespace fills up, so stopping the DDL gives you time to extend the tablespace size and prevent the loss of DDL capture. Managing tablespace sizing this way, however, requires frequent monitoring of the business applications and Extract to avoid business disruptions. As a best practice, make certain to size the tablespace appropriately in the first place, and set `AUTOEXTEND` to `ON` so that the tablespace does not fill up.

   > **WARNING:**
   >
   > Make a backup of the `params.sql` script before you edit it to preserve its original state.

6. Create a `GLOBALS` file (or edit it, if one exists).

   ```
   EDIT PARAMS ./GLOBALS
   ```

   > **Note:**
   >
   > `EDIT PARAMS` creates a simple text file. When you save the file after `EDIT PARAMS`, it is saved with the name `GLOBALS` in upper case, without a file extension, at the root of the Oracle GoldenGate directory. Do not alter the file name or location.

7. In the `GLOBALS` file, specify the name of the DDL schema by adding the following parameter to the `GLOBALS` file.

   ```
   GGSCHEMA schema_name
   ```

8. (Optional) To change the names of other objects listed in Table D-1 (page D-2), *the changes must be made now*, before proceeding with the rest of the installation.

Otherwise, you will need to stop Oracle GoldenGate DDL processing and reinstall the DDL objects. It is recommended that you accept the default names of the database objects. To change any name in Table D-1 (page D-2) (except the schema), do one or both of the following:

- Record all name changes in the `params.sql` script. Edit this script and change the appropriate parameters. Do not run this script.
- List the names shown in Table D-2 (page D-4) in the `GLOBALS` file. The correct parameters to use are listed in the **Parameter** column of the table.

Table D-2 GLOBALS Parameters for Changing DDL Object Names

Object	Parameter
Marker table	`MARKERTABLE` *`new_table_name`*[1]
History table	`DDLTABLE` *`new_table_name`*

[1] Do not qualify the name of any of these tables. The schema name for these table must be either the one that is specified with `GGSCHEMA` or the schema of the current user, if `GGSCHEMA` is not specified in `GLOBALS`.

9. To enable trigger-based DDL replication to recognize Oracle invisible indexes as unique identifiers, set the following parameter to `TRUE` in the `params.sql` script:

   ```
   define allow_invisible_index_keys = 'TRUE'
   ```

10. Save and close the `GLOBALS` file and the `params.sql` file.
11. Change directories to the Oracle GoldenGate installation directory.
12. Exit all Oracle sessions, including those of SQL*Plus, those of business applications, those of the Oracle GoldenGate processes, and those of any other software that uses Oracle. Prevent the start of any new sessions.
13. Run SQL*Plus and log in as a user that has `SYSDBA` privilege. This privilege is required to install the DDL trigger in the `SYS` schema, which is required by Oracle. All other DDL objects are installed in the schema that you created in 1 (page D-3).
14. Run the `marker_setup.sql` script. Supply the name of the Oracle GoldenGate schema when prompted, and then press **Enter** to execute the script. The script installs support for the Oracle GoldenGate DDL marker system.

    ```
    @marker_setup.sql
    ```

15. Run the `ddl_setup.sql` script. You are prompted to specify the name of the DDL schema that you configured in 1 (page D-3). (Note: `ddl_setup.sql` will fail if the tablespace for this schema is shared by any other users. It will not fail, however, if the default tablespace does not have `AUTOEXTEND` set to `ON`, the recommended setting.)

    ```
    @ddl_setup.sql
    ```

16. Run the `role_setup.sql` script. At the prompt, supply the DDL schema name. The script drops and creates the role that is needed for DDL synchronization, and it grants DML permissions on the Oracle GoldenGate DDL objects.

    ```
    @role_setup.sql
    ```

17. Grant the role that was created (default name is `GGS_GGSUSER_ROLE`) to all Oracle GoldenGate Extract users. You may need to make multiple grants if the processes have different user names.

```
GRANT role TO user;
```

18. Run the `ddl_enable.sql` script to enable the DDL trigger.

```
@ddl_enable.sql
```

To Install and Use the Optional Performance Tool

To improve the performance of the DDL trigger, make the `ddl_pin` script part of the database startup. It must be invoked with the Oracle GoldenGate DDL user name, as in:

```
@ddl_pin DDL_user
```

This script pins the PL/SQL package that is used by the trigger into memory. If executing this script from SQL*Plus, connect as `SYSDBA` from the Oracle GoldenGate installation directory. This script relies on the Oracle `dmbs_shared_pool` system package, so install that package before using `ddl_pin`.

E

Supporting Changes to XML Schemas

This appendix contains instructions for supporting changes to an XML schema. Both classic and integrated capture modes do not support the capture of changes made to an XML schema.

Topics:

E.1 Supporting RegisterSchema

`RegisterSchema` can be handled by registering the schema definition on both source and target databases before any table is created that references the XML schema.

E.2 Supporting DeleteSchema:

Issue `DeleteSchema` on the source database first. Once Replicat is caught up with the changes made to the source database, issue the `DeleteSchema` call on the target database.

E.3 Supporting CopyEvolve

The CopyEvolve procedure evolves, or changes, a schema and can modify tables by adding or removing columns. It can also be used to change whether or not XML documents are valid. Handling `CopyEvolve` requires more coordination. Use the following procedure if you are issuing `CopyEvolve` on the source database.

1. Quiesce changes to dependent tables on the source database.
2. Execute the `CopyEvolve` on the primary or source database.
3. Wait for Replicat to finish applying all of the data from those tables to the target database.
4. Stop Replicat.
5. Apply the `CopyEvolve` on the target database.
6. Restart Replicat.

F

Preparing DBFS for an Active-Active Configuration

This appendix contains steps to configure Oracle GoldenGate to function within an active-active bidirectional or multi-directional environment where Oracle Database File System (DBFS) is in use on both (or all) systems.
Topics:

F.1 Supported Operations and Prerequisites

Oracle GoldenGate for DBFS supports the following:

- Supported DDL (like `TRUNCATE` or `ALTER`) on DBFS objects except for `CREATE` statements on the DBFS objects. `CREATE` on DBFS must be excluded from the configuration, as must any schemas that will hold the created DBFS objects. The reason to exclude `CREATES` is that the metadata for DBFS must be properly populated in the SYS dictionary tables (which itself is excluded from Oracle GoldenGate capture by default).
- Capture and replication of DML on the tables that underlie the DBFS file system.

The procedures that follow assume that Oracle GoldenGate is configured properly to support active-active configuration. This means that it must be:

- Installed according to the instructions in this guide.
- Configured according to the instructions in the Oracle GoldenGate Windows and UNIX Administrator's Guide.

F.2 Applying the Required Patch

Apply the Oracle DBFS patch for bug-9651229 on both databases. To determine if the patch is installed, run the following query:

```
connect / as sysdba
select  procedure_name
from    dba_procedures
where   object_name = 'DBMS_DBFS_SFS_ADMIN'
and procedure_name = 'PARTITION_SEQUENCE';
```

The query should return a single row. Anything else indicates that the proper patched version of DBFS is not available on your database.

F.3 Examples Used in these Procedures

The following procedures assume two systems and configure the environment so that DBFS users on both systems see the same DBFS files, directories, and contents that are kept in synchronization with Oracle GoldenGate. It is possible to extend these concepts to support three or more peer systems.

F.4 Partitioning the DBFS Sequence Numbers

DBFS uses an internal sequence-number generator to construct unique names and unique IDs. These steps partition the sequences into distinct ranges to ensure that there are no conflicts across the databases. After this is done, further DBFS operations (both creation of new file systems and subsequent file system operations) can be performed without conflicts of names, primary keys, or IDs during DML propagation.

1. Connect to each database as sysdba.

   Issue the following query on each database.

   ```
   select last_number
   from dba_sequences
   where sequence_owner = 'SYS'
   and sequence_name = 'DBFS_SFS_$FSSEQ'
   ```

2. From this query, choose the maximum value of `LAST_NUMBER` across both systems, or pick a high value that is significantly larger than the current value of the sequence on either system.

3. Substitute this value ("`maxval`" is used here as a placeholder) in both of the following procedures. These procedures logically index each system as `myid=0` and `myid=1`.

   Node1

   ```
   declare
   begin
   dbms_dbfs_sfs_admin.partition_sequence(nodes => 2, myid => 0, newstart
   => :maxval);
   commit;
   end;
   /
   ```

   Node 2

   ```
   declare
   begin
   dbms_dbfs_sfs_admin.partition_sequence( nodes => 2, myid => 1, newstart
   => :maxval);
   commit;
   end;
   /
   ```

> **Note:**
>
> Notice the difference in the value specified for the `myid` parameter. These are the different index values.

For a multi-way configuration among three or more databases, you could make the following alterations:

- Adjust the maximum value that is set for `maxval` upward appropriately, and use that value on all nodes.
- Vary the value of `myid` in the procedure from 0 for the first node, 1 for the second node, 2 for the third one, and so on.

4. (Recommended) After (and only after) the DBFS sequence generator is partitioned, create a new DBFS file system on each system, and use only these file systems for DML propagation with Oracle GoldenGate. See Configuring the DBFS file system (page F-3).

> **Note:**
>
> DBFS file systems that were created before the patch for bug-9651229 was applied or before the DBFS sequence number was adjusted can be configured for propagation, but that requires additional steps not described in this document. If you must retain old file systems, open a service request with Oracle Support.

F.5 Configuring the DBFS file system

To replicate DBFS file system operations, use a configuration that is similar to the standard bidirectional configuration for DML.

- Use matched pairs of identically structured tables.
- Allow each database to have write privileges to opposite tables in a set, and set the other one in the set to read-only. For example:
  - Node1 writes to local table `t1` and these changes are replicated to `t1` on Node2.
  - Node2 writes to local table `t2` and these changes are replicated to `t2` on Node1.
  - On Node1, `t2` is read-only. On Node2, `t1` is read-only.

DBFS file systems make this kind of table pairing simple because:

- The tables that underlie the DBFS file systems have the same structure.
- These tables are modified by simple, conventional DML during higher-level file system operations.
- The DBFS ContentAPI provides a way of unifying the namespace of the individual DBFS stores by means of mount points that can be qualified as read-write or read-only.

The following steps create two DBFS file systems (in this case named FS1 and FS2) and set them to be read-write or read, as appropriate.

1. Run the following procedure to create the two file systems. (Substitute your store names for FS1 and FS2.)
2. Run the following procedure to give each file system the appropriate access rights. (Substitute your store names for FS1 and FS2.)

   In this example, note that on Node 1, store FS1 is read-write and store FS2 is read-only, while on Node 2 the converse is true: store FS1 is read-only and store FS2 is read-write.

   Note also that the read-write store is mounted as *local* and the read-only store is mounted as *remote*. This provides users on each system with an identical namespace and identical semantics for read and write operations. Local path names can be modified, but remote path names cannot.

Example F-1

```
declare
dbms_dbfs_sfs.createfile system('FS1');
dbms_dbfs_sfs.createfile system('FS2');

dbms_dbfs_content.registerStore('FS1',
'posix', 'DBMS_DBFS_SFS');
dbms_dbfs_content.registerStore('FS2',
'posix', 'DBMS_DBFS_SFS');
commit;
end;
/
```

Example F-2 Node 1

```
declare
dbms_dbfs_content.mountStore('FS1', 'local');
dbms_dbfs_content.mountStore('FS2', 'remote',
read_only => true);
commit;
end;
/
```

Example F-3 Node 2

```
declare
dbms_dbfs_content.mountStore('FS1', 'remote',
read_only => true);
dbms_dbfs_content.mountStore('FS2', 'local');
commit;
end;
/
```

F.6 Mapping Local and Remote Peers Correctly

The names of the tables that underlie the DBFS file systems are generated internally and dynamically. Continuing with the preceding example, there are:

- Two nodes (Node 1 and Node 2 in the example).
- Four stores: two on each node (FS1 and FS2 in the example).

- Eight underlying tables: two for each store (a table and a ptable). These tables must be identified, specified in Extract `TABLE` statements, and mapped in Replicat `MAP` statements.

1. To identify the table names that back each file system, issue the following query. (Substitute your store names for `FS1` and `FS2`.)

   The output looks like the following examples.

2. Identify the tables that are *locally read-write* to Extract by creating the following `TABLE` statements in the Extract parameter files. (Substitute your pluggable database names, schema names, and table names as applicable.)

3. Link changes on each remote file system to the corresponding local file system by creating the following `MAP` statements in the Replicat parameter files. (Substitute your pluggable database, schema and table names.)

   This mapping captures and replicates local read-write *source* tables to remote read-only peer tables:

   - file system changes made to `FS1` on Node 1 propagate to `FS1` on Node 2.
   - file system changes made to `FS2` on Node 2 propagate to `FS2` on Node1.

   Changes to the file systems can be made through the DBFS ContentAPI (package `DBMS_DBFS_CONTENT`) of the database or through `dbfs_client` mounts and conventional file systems tools.

   All changes are propagated in both directions.

   - A user at the virtual root of the DBFS namespace on each system sees identical content.
   - For mutable operations, users use the `/local` sub-directory on each system.
   - For read operations, users can use either of the `/local` or `/remote` sub-directories, depending on whether they want to see local or remote content.

Example F-4

```
select fs.store_name, tb.table_name, tb.ptable_name
from table(dbms_dbfs_sfs.listTables) tb,
table(dbms_dbfs_sfs.listfile systems) fs
where   fs.schema_name = tb.schema_name
and fs.table_name = tb.table_name
and fs.store_name in ('FS1', 'FS2')
;
```

Example F-5 Example output: Node 1 (Your Table Names Will Be Different.)

```
STORE NAME     TABLE_NAME     PTABLE_NAME
-------------  -------------  -------------
FS1            SFS$_FST_100   SFS$_FSTP_100
FS2            SFS$_FST_118   SFS$_FSTP_118
```

Example F-6 Example output: Node 2 (Your Table Names Will Be Different.)

```
STORE NAME     TABLE_NAME     PTABLE_NAME
-------------  -------------  -------------
FS1            SFS$_FST_101   SFS$_FSTP_101
FS2            SFS$_FST_119   SFS$_FSTP_119
```

Example F-7 Node1

```
TABLE [container.]schema.SFS$_FST_100
TABLE [container.]schema.SFS$_FSTP_100;
```

Example F-8 Node2

```
TABLE [container.]schema.SFS$_FST_119
TABLE [container.]schema.SFS$_FSTP_119;
```

Example F-9 Node1

```
MAP [container.]schema.SFS$_FST_119, TARGET [container.]schema.SFS$_FST_118;
MAP [container.]schema.SFS$_FSTP_119, TARGET [container.]schema.SFS$_FSTP_118
```

Example F-10 Node2

```
MAP [container.]schema.SFS$_FST_100, TARGET [container.]schema.SFS$_FST_101;MAP
[container.]schema.SFS$_FSTP_100, TARGET [container.]schema.SFS$_FSTP_101;
```

G

Oracle GoldenGate Installed Components

This appendix describes the programs, directories, and other components created or used by the Oracle GoldenGate software in the Oracle GoldenGate installation directory.

Topics:

G.1 Oracle GoldenGate Programs and Utilities

This section describes programs installed in the root Oracle GoldenGate installation directory.

> **Note:**
>
> Some programs may not exist in all installations. For example, if only capture or delivery is supported by Oracle GoldenGate for your platform, the extract or replicat program will not be installed, respectively. Likewise, special files might be installed to support a specific database.

Table G-1 Oracle GoldenGate Installed Programs and Utilities

Program	Description
`convchk`	Converts checkpoint files to a newer version.
`convprm`	Converts parameter files that do not use SQL-92 rules for quoted names and literals to updated parameter files that use SQL-92 rules. SQL-92 format for quoted object names and literals was introduced as the default with version 12c of Oracle GoldenGate.
`defgen`	Generates data definitions and is referenced by Oracle GoldenGate processes when source and target tables have dissimilar definitions.
`emsclnt`	Sends event messages created by Collector and Replicat on Windows or UNIX systems to EMS on NonStop systems.
`extract`	Performs capture from database tables or transaction logs or receives transaction data from a vendor access module.
`ggmxinstall`	Oracle GoldenGate installation script for the SQL/MX database.

Table G-1 (Cont.) Oracle GoldenGate Installed Programs and Utilities

Program	Description
`ggcmd`	Associated program of `ggsci`. Launches and monitors external applications, such as the JAGENT of Oracle GoldenGate Monitor. Integrates those applications into the GGSCI environment.
`ggsci`	User interface to Oracle GoldenGate for issuing commands and managing parameter files.

`ggsmgr.jcl`
`ggsmgr.proc`
`ggsmgrst.jcl`
`ggsmgrst.proc`
	Start the Oracle GoldenGate Manager process from a batch job or the operator console on a z/OS system. Installed to support DB2 z/OS databases.

`install`	Installs Oracle GoldenGate as a Windows service and provides other Windows-based service options.
`keygen`	Generates data-encryption keys.
`logdump`	A utility for viewing and saving information stored in extract trails or files.
`mgr`	(Manager) Control process for resource management, control and monitoring of Oracle GoldenGate processes, reporting, and routing of requests through the GGSCI interface.
`oggerr`	Manages Oracle GoldenGate error messages.
`replicat`	Applies data to target database tables.
`reverse`	A utility that reverses the order of transactional operations, so that Replicat can be used to back out changes from target tables, restoring them to a previous state.
`server`	The Collector process, an Extract TCP/IP server collector that writes data to remote trails.
`vamserv`	Started by Extract to read the TMF audit trails generated by TMF-enabled applications. Installed to support the NonStop SQL/MX database.

G.2 Oracle GoldenGate Subdirectories

This Section describes the subdirectories of the Oracle GoldenGate installation directory and their contents.

> **Note:**
>
> Some directories may not exist in all installations.

Table G-2 Oracle GoldenGate Installed Subdirectories

Directory	Description
`br`	Contains the checkpoint files for the bounded recover feature.

Table G-2 (Cont.) Oracle GoldenGate Installed Subdirectories

Directory	Description
`cfg`	Contains the property and XML files that are used to configure Oracle GoldenGate Monitor.
`dirdb`	Contains the data store that is used to persist information that is gathered from an Oracle GoldenGate instance for use by the Oracle GoldenGate Monitor application or within Oracle Enterprise Manager.

`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 Oracle GoldenGate format.
	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.
	Do not edit these files.
	Examples:
	`ext1.cpe`
	`rep1.cpr`

`dircrd`	Contains credential store files.

`dirdat`	The default location for Oracle GoldenGate trail files and extract files that are created by Extract processes to store extracted data for further processing by the Replicat process or another application or utility. Written in internal Oracle GoldenGate format.
	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).
	Do not edit these 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 Oracle GoldenGate technical support. Example: `defs.dat`
`dirdmp`	Contains trace, or dump, files that support the internal activity logging mechanism.
`dirjar`	Contains the Java executable files that support Oracle GoldenGate Monitor.

Table G-2 (Cont.) Oracle GoldenGate Installed Subdirectories

Directory	Description

`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 number, and the process ID.
	Do not edit these files.
	Examples:
	`mgr.pcm`
	`ext.pce`

`dirprm`	The default location for Oracle GoldenGate parameter files created by Oracle GoldenGate users to store run-time parameters for Oracle 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 Oracle GoldenGate parameter values after stopping the process. They can be edited directly from a text editor or by using the `EDIT PARAMS` command in GGSCI.
	Examples:
	`defgen.prm`
	`finance.prm`

`dirrec`	Not used by Oracle GoldenGate.

`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`	Contains training scripts and any user-created SQL scripts that support Oracle GoldenGate.
`dirtmp`	The default location for storing transaction data when the size exceeds the memory size that is allocated for the cache manager. Do not edit these files.
`dirwlt`	Contains Oracle GoldenGate wallet files.
`UserExitExamples`	Contains sample files to help with the creation of user exits.

G.3 Other Oracle GoldenGate Files

This section describes other files, templates, and objects created or installed in the root Oracle GoldenGate installation directory.

> **Note:**
>
> Some files may not be installed in your environment, depending on the database and OS platform.

Table G-3 Other Oracle GoldenGate Installed Files

Component	Description
`bcpfmt.tpl`	Template for use with Replicat when creating a run file for the Microsoft BCP/DTS bulk-load utility.
`bcrypt.txt`	Blowfish encryption software license agreement.
`cagent.dll`	Contains the Windows dynamic link library for the Oracle GoldenGate Monitor C sub-agent.
`category.dll`	Windows dynamic link library used by the `INSTALL` utility.
`chkpt_`*`db`*`_create.sql`	Script that creates a checkpoint table in the local database. A different script is installed for each database type.
`db2cntl.tpl`	Template for use with Replicat when creating a control file for the IBM `LOADUTIL` bulk-load utility.
`ddl_cleartrace.sql`	Script that removes the DDL trace file. (Oracle installations)
`ddl_ddl2file.sql`	Script that saves DDL from the marker table to a file.
`ddl_disable.sql`	Script that disables the Oracle GoldenGate DDL trigger. (Oracle installations)
`ddl_enable.sql`	Script that enables the Oracle GoldenGate DDL trigger. (Oracle installations)
`ddl_filter.sql`	Script that supports filtering of DDL by Oracle GoldenGate. This script runs programmatically; do not run it manually.
`ddl_nopurgeRecyclebin.sql`	Empty script file for use by Oracle GoldenGate support staff.
`ddl_ora11.sql` `ddl_ora12.sql`	Scripts that run programmatically as part of Oracle GoldenGate DDL support; do not run these scripts.
`ddl_pin.sql`	Script that pins DDL tracing, the DDL package, and the DDL trigger for performance improvements. (Oracle installations)
`ddl_purgeRecyclebin.sql`	Script that purges the Oracle recycle bin in support of the DDL replication feature.
`ddl_remove.sql`	Script that removes the DDL extraction trigger and package. (Oracle installations)
`ddl_session.sql` ddl_session1.sql	Supports the installation of the Oracle DDL objects. This script runs programmatically; do not run it manually.
`ddl_setup.sql`	Script that installs the Oracle GoldenGate DDL extraction and replication objects. (Oracle installations)
`ddl_status.sql`	Script that verifies whether or not each object created by the Oracle GoldenGate DDL support feature exists and is functioning properly. (Oracle installations)

Table G-3 (Cont.) Other Oracle GoldenGate Installed Files

Component	Description
`ddl_staymetadata_off.sql` `ddl_staymetadata_on.sql`	Scripts that control whether the Oracle DDL trigger collects metadata. This script runs programmatically; do not run it manually.
`ddl_trace_off.sql` `ddl_trace_on.sql`	Scripts that control whether DDL tracing is on or off.
`ddl_tracelevel.sql`	Script that sets the level of tracing for the DDL support feature. (Oracle installations)
`debug files`	Debug text files that may be present if tracing was turned on.
`demo_`*`db_scriptname`*`.sql` `demo_`*`more_db_scriptname`*`.sql`	Scripts that create and populate demonstration tables for use with tutorials and basic testing.
`.dmp files`	Dump files created by Oracle GoldenGate processes for tracing purposes.
`ENCKEYS`	User-created file that stores encryption keys. Written in external ASCII format.
`exitdemo.c`	User exit example.
`exitdemo_utf16.c`	User exit example that demonstrates how to use UTF16 encoded data in the callback structures for information exchanged between the user exit and the process.
`freeBSD.txt`	License agreement for FreeBSD.
`ggmessage.dat`	Data file that contains error, informational, and warning messages that are returned by the Oracle GoldenGate processes. The version of this file is checked upon process startup and must be identical to that of the process in order for the process to operate.
`ggserr.log`	File that logs processing events, messages, errors, and warnings generated by Oracle GoldenGate.
`ggsmsg.dll`	Windows dynamic link library used by the `install` program.
`GLOBALS`	User-created file that stores parameters applying to the Oracle GoldenGate instance as a whole.
`help.txt`	Help file for the GGSCI command interface.

`icudtxx.dll`
`icuinxx.dll`
`icuucxx.dll`
	Windows shared libraries for International Components for Unicode, where *xx* is the currently used version.

`jagent.bat`	Windows batch file for the Java Agent for Oracle GoldenGate Monitor.
`jagent.log` `jagentjni.log`	Log files for the Oracle GoldenGate Monitor Agent.
`jagent.sh`	UNIX shell script for the Java Agent for Oracle GoldenGate Monitor
`LGPL.txt`	Lesser General Public License statement. Applies to free libraries from the Free Software Foundation.

Table G-3 (Cont.) Other Oracle GoldenGate Installed Files

Component	Description
libodbc.so	ODBC file for Ingres 2.6 on UNIX.
libodbc.txt	License agreement for libodbc.so.
libxml2.dll	Windows dynamic link library containing the XML library for the Oracle GoldenGate XML procedures.
libxml2.txt	License agreement for libxml2.dll .
marker.hist	File created by Replicat if markers were passed from a NonStop source system.
marker_remove.sql	Script that removes the DDL marker table. (Oracle installations)
marker_setup.sql	Script that installs the Oracle GoldenGate DDL marker table. (Oracle installations)
marker_status.sql	Script that confirms successful installation of the DDL marker table. (Oracle installations)
notices.txt	Third-party software license file.
odbcinst.ini	Ingres 2.6 on UNIX ODBC configuration file.
params.sql	Script that contains configurable parameters for DDL support. (Oracle installations)
pthread-win32.txt	License agreement for pthread-VC.dll .
pthread-VC.dll	POSIX threads library for Microsoft Windows.
prvtclkm.plb	Supports the replication of Oracle encrypted data.
pw_agent_util.bat pw_agent_util.sh	Script files that support the Oracle GoldenGate Monitor Agent.
role_setup.sql	Script that creates the database role necessary for Oracle GoldenGate DDL support. (Oracle installations)
sampleodbc.ini	Sample ODBC file for Ingres 2.6 on UNIX.
sqlldr.tpl	Template for use with Replicat when creating a control file for the Oracle SQL*Loader bulk-load utility.
start.prm stop.prm	z/OS paramlib members to start and stop the Manager process.
startmgr stopmgr	z/OS UNIX System Services scripts to start the Manager process from GGSCI.
startmgrcom stopmgrcom	z/OS system input command for the Manager process.
tcperrs	File containing user-defined instructions for responding to TCP/IP errors.
usrdecs.h	Include file for user exit API.
xerces-c_2_8.dll	Apache XML parser library.
zlib.txt	License agreement for zlib compression library.

G.4 Oracle GoldenGate Checkpoint Table

When database checkpoints are being used, Oracle GoldenGate creates a checkpoint table with a user-defined name in the database upon execution of the `ADD CHECKPOINTTABLE` command, or a user can create the table by using the `chkpt_db_create.sql` script (where *db* is an abbreviation of the type of database that the script supports). For a description of this table, see *Administering Oracle GoldenGate for Windows and UNIX*.