Oracle® GoldenGate

Installing and Configuring Oracle GoldenGate for Teradata

12*c* (12.1.2)

E29290-02

May 2015

Contains system requirements, installation, and setup instructions for Oracle GoldenGate for the Teradata database.

ORACLE®

Oracle GoldenGate Installing and Configuring Oracle GoldenGate for Teradata, 12*c* (12.1.2)

E29290-02

Contents

Preface vii

Audience vii
Documentation Accessibility vii
..... vii
Related Documents vii
Conventions viii

1 System Requirements and Preinstallation Instructions

1.1 Overview of Oracle GoldenGate for Teradata 1-1
1.2 What this Documentation Provides 1-1
1.3 Verifying Certification and System Requirements 1-2
1.4 Supported Platforms for a Replication Server 1-2
1.5 Operating System Requirements 1-2
1.5.1 Memory Requirements 1-2
1.5.2 Disk Requirements 1-3
1.5.3 Relay Services Gateway (RSG) vprocs 1-4
1.5.4 Network 1-4
1.5.5 Operating System Privileges 1-4
1.5.6 Console 1-4
1.5.7 Other Programs 1-4
1.6 Database Requirements 1-5
1.6.1 Database Configuration 1-5
1.6.2 Database User for Oracle GoldenGate Processes 1-5
1.7 Supported Teradata Data Types 1-6
1.7.1 Limitations of Support for Numeric Data Types 1-7
1.7.2 Limitations of Support for Single-byte Character Data Types 1-7
1.7.3 Conditions and Limitations of Support for Multi-byte Character Data 1-7
1.7.4 Limitations of Support for Binary Data Types 1-8
1.7.5 Limitations of Support for Large Object Data Types 1-8
1.7.6 Limitations of Support for Date Data Types 1-8
1.7.7 Limitations of Support for IDENTITY Data Types 1-9
1.8 Supported Objects and Operations for Teradata 1-9
1.8.1 DML 1-9
1.8.2 DDL 1-9

2 Installing Oracle GoldenGate

2.1 Installation Overview 2-1
2.2 Downloading Oracle GoldenGate 2-1
2.3 Setting Library Paths for Dynamic Builds on UNIX 2-2
2.4 Preparing to Install Oracle GoldenGate within a Cluster 2-4
2.4.1 Deciding Where to Install Oracle GoldenGate Binaries and Files in the Cluster 2-4
2.5 Installing Oracle GoldenGate on Linux and UNIX 2-4
2.6 Installing Oracle GoldenGate on Windows 2-5
2.6.1 Installing Oracle GoldenGate into a Windows Cluster 2-5
2.6.2 Installing the Oracle GoldenGate Files 2-5
2.6.3 Specifying a custom Manager name 2-6
2.6.4 Installing Manager as a Windows Service 2-6
2.7 Integrating Oracle GoldenGate into a Cluster 2-8
2.7.1 General Requirements in a Cluster 2-8
2.7.2 Adding Oracle GoldenGate as a Windows Cluster Resource 2-8

3 Preparing the System for Oracle GoldenGate

3.1 Preparing Tables for Processing 3-1
3.1.1 Disabling Triggers and Cascade Constraints 3-1
3.1.2 Assigning Row Identifiers 3-2
3.1.2.1 How Oracle GoldenGate Determines the Kind of Row Identifier to Use 3-2
3.1.2.2 Using KEYCOLS to Specify a Custom Key 3-2
3.2 Creating a Teradata Replication Group 3-2
3.3 Activating DDL Capture by the Teradata TAM 3-3
3.4 Disabling the Capture of DDL 3-4
3.5 Configuring the TAM Initialization File 3-5

4 Configuring Oracle GoldenGate

4.1 Configuring Oracle GoldenGate in Maximum Protection Mode 4-1
4.1.1 Recommended Maximum Protection Configuration 4-1
4.1.2 Configuring Extract in Maximum Protection Mode 4-2
4.1.3 Configuring Replicat to Support Maximum Protection Mode 4-4
4.2 Configuring Oracle GoldenGate in Maximum Performance Mode 4-5
4.2.1 Recommended Maximum Performance Configuration 4-5
4.2.2 Configuring Extract in Maximum Performance Mode 4-6
4.2.3 Configuring Replicat to Support Maximum Performance Mode 4-7
4.3 Additional Oracle GoldenGate Configuration Guidelines 4-8
4.3.1 Configuring DDL Support 4-8
4.3.2 Handling Massive Update and Delete Operations 4-8
4.3.3 Preventing Multiple Connections 4-8
4.3.4 Performing Initial Synchronization 4-8

5 Configuring DDL Synchronization for a Teradata Database

5.1 About this Chapter 5-1
5.2 Overview of DDL Synchronization 5-2
5.3 Limitations of Oracle GoldenGate DDL Support 5-2

5.3.1 DDL Statement Length ... 5-2
5.3.2 Supported Topologies ... 5-2
5.3.3 Filtering, Mapping, and Transformation ... 5-2
5.3.4 DDL Response Time ... 5-3
5.4 Configuration Guidelines for DDL Support ... 5-3
5.4.1 Database Privileges ... 5-3
5.4.2 Parallel Processing ... 5-3
5.4.3 DDL and DML in Data Pumps and VAM-sort Extracts ... 5-3
5.4.4 Object Names ... 5-4
5.4.5 Data Definitions ... 5-4
5.4.6 Initial Synchronization ... 5-4
5.5 Understanding DDL Scopes ... 5-4
5.5.1 Mapped Scope ... 5-5
5.5.1.1 Mapping ALTER INDEX ... 5-6
5.5.2 Unmapped Scope ... 5-6
5.5.3 Other Scope ... 5-6
5.6 Enabling DDL Support ... 5-7
5.7 Filtering DDL Replication ... 5-7
5.7.1 Combining DDL Parameter Options ... 5-10
5.7.2 DDL EXCLUDE ALL ... 5-11
5.8 How Oracle GoldenGate Handles Derived Object Names ... 5-11
5.8.1 Mapping Derived Objects ... 5-11
5.8.1.1 MAP Exists forBbase Object, But Not Derived Object ... 5-12
5.8.1.2 MAP Exists for Base and Derived Objects ... 5-12
5.8.1.3 MAP Exists for Derived Object, But Not Base Object ... 5-12
5.8.2 New Tables as Derived Objects ... 5-13
5.8.2.1 RENAME ... 5-13
5.8.2.2 CREATE TABLE AS SELECT ... 5-13
5.8.3 Disabling the Mapping of Derived Objects ... 5-14
5.9 Using DDL String Substitution ... 5-15
5.10 How DDL is Evaluated for Processing ... 5-15
5.10.1 How Extract Evaluates DDL ... 5-15
5.10.2 How Replicat Evaluates DDL ... 5-16
5.11 Handling DDL Processing Errors ... 5-17
5.12 Viewing DDL Report Information ... 5-17
5.12.1 Extract DDL Reporting ... 5-17
5.12.2 Replicat DDL Reporting ... 5-18
5.12.3 Statistics in the Process Reports ... 5-18
5.13 Tracing DDL Processing ... 5-19

6 Modifying Objects in the Oracle GoldenGate Configuration

6.1 Deleting an Extract Group ... 6-1
6.2 Adding a Table to an Existing Extract Group ... 6-1
6.3 Moving a Table to a New Extract Group ... 6-2
6.4 Modifying Columns of a Table ... 6-3

7 Uninstalling Oracle GoldenGate

7.1 Uninstalling Oracle GoldenGate from Linux or UNIX 7-1
7.2 Uninstalling Oracle GoldenGate from Windows (Non-cluster) 7-2
7.3 Uninstalling Oracle GoldenGate from Windows Cluster 7-2

A Oracle GoldenGate Installed Components

A.1 Oracle GoldenGate Programs and Utilities A-1
A.2 Oracle GoldenGate Subdirectories A-2
A.3 Other Oracle GoldenGate Files A-4
A.4 Oracle GoldenGate Checkpoint Table A-7

Preface

This guide helps you get started with installing Oracle GoldenGate in a Teradata database environment and configuring the Teradata database to support Oracle GoldenGate capture and delivery.

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.

Access to Oracle Support

Oracle customers that 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 Documents

The Oracle GoldenGate documentation set includes the following components:

Windows, UNIX, and Linux Platforms

- *Installing and Configuring Oracle GoldenGate for DB2 for i*
- *Installing and Configuring Oracle GoldenGate for DB2 LUW*
- *Installing and Configuring Oracle GoldenGate for DB2 z/OS*
- *Installing and Configuring Oracle GoldenGate for Informix*
- *Installing and Configuring Oracle GoldenGate for MySQL*
- *Installing and Configuring Oracle GoldenGate for NonStop SQL/MX*
- *Installing and Configuring Oracle GoldenGate for SQL Server*

- *Installing and Configuring Oracle GoldenGate for Oracle TimesTen*
- *Installing and Configuring Oracle GoldenGate for Oracle Database*
- *Installing and Configuring Oracle GoldenGate for Sybase*
- *Installing and Configuring Oracle GoldenGate for Teradata*
- *Administering Oracle GoldenGate for Windows and UNIX*
- *Reference for Oracle GoldenGate for Windows and UNIX*
- *Logdump Reference for Oracle GoldenGate*
- *Upgrading Oracle GoldenGate for Windows and UNIX*
- *Error Messages Reference for Oracle GoldenGate for Windows and UNIX*

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*][*option1* | *option2*]. Multiple options within an optional element are separated by a pipe symbol, for example:

1

System Requirements and Preinstallation Instructions

This chapter contains the requirements for the system and database resources that support Oracle GoldenGate. It contains the following sections:

- Section 1.1, "Overview of Oracle GoldenGate for Teradata"
- Section 1.2, "What this Documentation Provides"
- Section 1.3, "Verifying Certification and System Requirements"
- Section 1.4, "Supported Platforms for a Replication Server"
- Section 1.5, "Operating System Requirements"
- Section 1.6, "Database Requirements"
- Section 1.7, "Supported Teradata Data Types"
- Section 1.8, "Supported Objects and Operations for Teradata"

1.1 Overview of Oracle GoldenGate for Teradata

Oracle GoldenGate supports the replication of data as follows:

- between a Teradata source database (known as a *source server*) and a Teradata target database (known as a *subscriber server*).
- between Teradata databases and other supported database platforms.

In addition, Oracle GoldenGate replicates DDL operations between identical Teradata source and subscriber servers.

Oracle GoldenGate receives transactional changes or table-copy operations from the Teradata Change Data Capture (CDC) facility and transmits them to the subscriber over ODBC. Communication between the CDC and Oracle GoldenGate is managed by the Teradata Access Module (TAM).

Oracle GoldenGate for Teradata supports the filtering, mapping, and transformation of data unless noted otherwise in this documentation.

1.2 What this Documentation Provides

This documentation contains information that is specific to the setup of the Oracle GoldenGate solution within a Teradata environment. It assumes that the reader has a fundamental knowledge of the Teradata database and the Teradata Replication Solutions. It also assumes that the following have been configured properly:

- Relay Services Gateway (RSG)
- Change Data Capture (CDC)
- Teradata Access Module (TAM)
- Replication groups

To configure replication for the Teradata database, see the *Teradata Replication Solutions* documentation from Teradata Corporation.

1.3 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.

1.4 Supported Platforms for a Replication Server

In a Teradata environment, you install Oracle GoldenGate on a server that is separate from the one where the Teradata source and target databases are installed. This machine will be the replication server and must be a platform that is supported by Oracle GoldenGate for the Teradata database. To find out which Oracle GoldenGate builds are available for a specific combination of database version and operating system, log onto `http://support.oracle.com` and select the Certifications tab. For assistance, click Tips for Finding Certifications. An e-mail and password are required to enter this site.

Some notes about choosing a replication server or servers:

- The replication server can exist in the same location as the source or target server, or it can be remote from one or both. For replication between Teradata systems in remote locations, install the Oracle GoldenGate Extract process on a replication server at the source location, and install the Oracle GoldenGate Replicat process on a different replication server at the target location.
- If possible, install Oracle GoldenGate on a multi-node cluster server to minimize the impact of any Oracle GoldenGate outages that are caused by server failure.

For additional configuration considerations, consult the *Teradata Replication Services Using Oracle GoldenGate* documentation from Teradata Corporation before installing Oracle GoldenGate.

1.5 Operating System Requirements

This section outlines the operating system resources that are necessary to support Oracle GoldenGate. These resources may apply to the database servers, the replication server(s), or all servers.

1.5.1 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*.

1.5.2 Disk Requirements

The recommended hardware configuration for the Oracle GoldenGate replication server is:

- Four 300-GB disks
- 4 dual-core CPUs
- 8 GB of RAM

Assign the following free disk space:

- To determine the size of the Oracle GoldenGate download file, view the Size column before downloading your selected build from Oracle Software Delivery Cloud. The value shown is the size of the files in compressed form. The size of the expanded Oracle GoldenGate installation directory will be significantly larger on disk. For more information, see Section 2.2, "Downloading Oracle GoldenGate."
- Allow at least an additional 1 GB of disk space on any system that hosts Oracle GoldenGate trails, which are files that contain the working data. 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*.
- To install Oracle GoldenGate into a cluster environment, install the Oracle GoldenGate binaries and files on a shared file system that is available to all cluster nodes.

1.5.3 Relay Services Gateway (RSG) vprocs

Replication tasks run on RSG vprocs on the source database server for connections with the replication server. The connection implements the TCP/IP protocol. As of Teradata V12, each system node can have one RSG.

1.5.4 Network

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

- Configure the systems that are involved with Oracle GoldenGate to use TCP/IP 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 Manager process.
- Configure your firewalls to accept connections through the Oracle GoldenGate ports.

1.5.5 Operating System Privileges

The Manager process requires an operating system user that has privileges to control Oracle GoldenGate processes and to read, write, and purge files and subdirectories in the Oracle GoldenGate directory. The Extract and Replicat processes require privileges to access the database.

1.5.6 Console

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.

1.5.7 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 GoldenGate fully supports virtual machine environments created with any virtualization software on any platform. When installing Oracle GoldenGate into a virtual machine environment, select a build that matches the database and the operating system of the virtual machine, not the host system.

1.6 Database Requirements

This section contains Oracle GoldenGate requirements that are specific to the Teradata database.

1.6.1 Database Configuration

Follow these requirements for database configuration:

- Install an appropriate ODBC (Open Database Connectivity) driver for the database version that you are using. Oracle GoldenGate supports database versions 13.10, 14.00 and 14.10.x.
- Configure ODBC on each source and target system, including the creation of a data source name (DSN). A DSN stores information about how to connect to the database. See the *ODBC Driver for Teradata User Guide* at http://www.info.teradata.com/ for instructions.
- Create Teradata replication groups for the source tables. For instructions, see the *Teradata Replication Services Using Oracle GoldenGate* documentation, at http://www.info.teradata.com.

1.6.2 Database User for Oracle GoldenGate Processes

Follow these requirements for the database user for Oracle GoldenGate processes:

- Create a database user that is dedicated to Oracle GoldenGate. It can be the same user for all of the Oracle GoldenGate processes that must connect to a database:
  - Extract (source database)
  - Replicat (target database)
  - The DEFGEN utility (source or target database)
- 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, the Oracle GoldenGate database user.
- If using Oracle GoldenGate to capture from a source Teradata database, issue the following security grants to the Extract database user.

```
GRANT SELECT ON DBC.REPGROUP TO user;
GRANT SELECT ON DBC.TVM TO user;
GRANT SELECT ON DBC.DBASE TO user;
GRANT SELECT ON DBC.ERRORMSGS TO user;
GRANT SELECT ON DBC.TVFIELDS TO user;
GRANT SELECT ON DBC.INDEXES TO user;
GRANT SELECT ON DBC.INDOUBTRESLOG TO user;
GRANT REPLCONTROL TO user;
GRANT ALL ON database TO user;
GRANT ALL ON SYSUDTLIB TO user WITH GRANT OPTION;
```

- If using Oracle GoldenGate to replicate to a target Teradata database, grant `SELECT`, `INSERT`, `UPDATE`, and `DELETE` on all of the target tables to the Replicat database user.

1.7 Supported Teradata Data Types

Table 1–1 shows the Teradata data types that Oracle GoldenGate supports. Any limitations or conditions that apply follow this table.

Table 1–1 Supported Data Types by Oracle GoldenGate, Per Teradata Version

Data type	v12	v13	v13.1
`BLOB`	No	Yes	Yes
`BYTEINT`	Yes	Yes	Yes
`VARBYTE`	Yes	Yes	Yes
`BIGINT`	Yes	Yes	Yes
`BYTEINT`	Yes	Yes	Yes
`DATE`	Yes	Yes	Yes
`DECIMAL - 18 and under`	Yes	Yes	Yes
`DECIMAL - 19 to 38`	No	Yes	Yes
`DOUBLE PRECISION`	Yes	Yes	Yes
`FLOAT`	Yes	Yes	Yes
`INTEGER`	Yes	Yes	Yes
`NUMERIC - 18 and under`	Yes	Yes	Yes
`NUMERIC - 19 to 38`	No	Yes	Yes
`REAL`	Yes	Yes	Yes
`SMALLIINT`	Yes	Yes	Yes
`TIME`	Yes	Yes	Yes
`TIMESTAMP`	Yes	Yes	Yes
`INTERVAL`	Yes	Yes	Yes
`INTERVAL DAY`	Yes	Yes	Yes
`INTERVAL DAY TO HOUR`	Yes	Yes	Yes
`INTERVAL DAY TO MINUTE`	Yes	Yes	Yes
`INTERVAL DAY TO SECOND`	Yes	Yes	Yes
`INTERVAL HOUR`	Yes	Yes	Yes
`INTERVAL HOUR TO MINUTE`	Yes	Yes	Yes
`INTERVAL HOUR TO SECOND`	Yes	Yes	Yes
`INTERVAL MINUTE`	Yes	Yes	Yes

Table 1–1 (Cont.) Supported Data Types by Oracle GoldenGate, Per Teradata Version

Data type	v12	v13	v13.1
INTERVAL MINUTE TO SECOND	Yes	Yes	Yes
INTERVAL MONTH	Yes	Yes	Yes
INTERVAL SECOND	Yes	Yes	Yes
INTERVAL YEAR	Yes	Yes	Yes
INTERVAL YEAR TO MONTH	Yes	Yes	Yes
CHAR	Yes	Yes	Yes
CLOB	No	Yes	Yes
CHAR VARYING	Yes	Yes	Yes
LONG VARCHAR	Yes	Yes	Yes
VARCHAR	Yes	Yes	Yes
GRAPHIC	Yes	Yes	Yes
LONG VARGRAPHIC	Yes	Yes	Yes
VARGRAPHIC	Yes	Yes	Yes
PERIOD (DATE)	No	Yes	Yes
PERIOD (TIME)	No	Yes	Yes
PERIOD (TIMESTAMP)	No	Yes	Yes
UDT	No	Yes	Yes

1.7.1 Limitations of Support for Numeric Data Types

Numeric data types are fully supported between Teradata source and target databases. When replicating these data types from a different type of database to Teradata, truncation can occur if the source database supports a higher precision that Teradata does.

The support of 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.

1.7.2 Limitations of Support for Single-byte Character Data Types

Single-byte character types are fully supported within a single-byte Latin character set between a Teradata source and Teradata targets, and between other databases and Teradata. A `VARCHAR` or `CHAR` column cannot have more than 32k-1 bytes. If using UTF-16, this is 16k-2 characters.

1.7.3 Conditions and Limitations of Support for Multi-byte Character Data

Conditions and limitations of support for multi-byte character data are as follows:

- Install Oracle GoldenGate on a Windows or Linux replication server.
- Use the Teradata ODBC driver version 12.0.0.x or later.
- Do not use filtering, mapping, and transformation for multi-byte data types.

- Source Teradata tables can contain only CHAR, VARCHAR, INTEGER, SMALLINT, DATE, TIME, and TIMESTAMP columns. No other data types can be replicated while multi-byte data is being replicated.
- A CHAR or VARCHAR column cannot contain more than 32k-1 bytes. If using UTF-16, these columns cannot contain more than 16k-2 characters.
- Set the ODBC driver and the Teradata Access Module (TAM) to the UTF-16 character set in the initialization file.
- When creating Replicat groups, use the NODBCHECKPOINT option with the ADD REPLICAT command. The Replicat database checkpointing feature does not support an ODBC driver that is set to the UTF-16 character set. Checkpoints will be maintained in the checkpoint file on disk.
- To support Unicode on Linux, Oracle GoldenGate must be configured in pass-through mode, and the ODBC Teradata Linux driver must be version 12.0.0.7 or higher, or 13.0.0.5 or higher. For more information about pass-through mode, see the PASSTHRU parameter in *Reference for Oracle GoldenGate for Windows and UNIX*.

1.7.4 Limitations of Support for Binary Data Types

No limitations. These data types are supported between a Teradata source and Teradata targets, and between other source databases and Teradata targets.

1.7.5 Limitations of Support for Large Object Data Types

The following are limitations of support for large object data types.

- To replicate UDTs, the target database must be Teradata Database 12.00.00.01 or greater or later.
- To replicate large objects from other databases to Teradata, use Teradata ODBC driver version 12.0 or higher on the target system. The target must support large objects that are delivered by ODBC.
- Enable the UseNativeLOBSupport flag in the ODBC configuration file. See the Teradata ODBC documentation.

1.7.6 Limitations of Support for Date Data Types

The following are limitations of support for date data types:

- Date types are fully supported between Teradata source and Teradata target databases. Additionally, INTERVAL is supported between Teradata and Oracle if the size of the target column is equal to, or greater than, that of the source.
- DATE, TIME, and TIMESTAMP are fully supported when replicated from a different type of source database to Teradata.
- TIME with TIMESZONE, TIMESTAMP with TIMEZONE, and INTERVAL are not supported from a different type of source database to Teradata.
- 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 timezone, conversion may add or subtract hours, which can cause the timestamp to exceed the lower or upper supported limit.
- Oracle GoldenGate does not support negative dates.

1.7.7 Limitations of Support for IDENTITY Data Types

`IDENTITY` must be configured as `GENERATED BY DEFAULT AS IDENTITY` on the target to enable the correct value to be inserted by Replicat. To include `IDENTITY` in a bi-directional replication configuration, the ranges of the values defined on the source and target systems must be disjoint, for example odd on one and even on the other.

1.8 Supported Objects and Operations for Teradata

This section lists the data operations and database objects that Oracle GoldenGate supports.

1.8.1 DML

Oracle GoldenGate supports the maximum number of columns per table that is supported by the database.

1.8.2 DDL

A Teradata DDL statement can be replicated when it satisfies one of the following conditions:

- The DDL statement affects a table that is a member of a replication group.
- The DDL statement matches a user-defined replication rule.
- The DDL statement changes certain properties of a replication group.

Table 1–2 shows the Teradata DDL operations that Oracle GoldenGate supports. Statement lengths of up to 2 MB in length are supported. At least TAM 13.0 is required, and both source and target databases must be Teradata Database 13.0 or later.

Table 1–2 Supported Teradata DDL

Operations	Object
CREATE	TABLE *table name*[1]
	GLOBAL TEMPORARY TABLE *table name*[2]
	[RECURSIVE] VIEW *view name*
	MACRO *macro name*
	HASH INDEX *index name*
	JOIN INDEX *index name*
	TRIGGER *trigger name*
ALTER	TABLE
DROP	TABLE *table name*
	VIEW *view name*
	MACRO *macro name*
	HASH INDEX *index name*
	JOIN INDEX *index name*
	TRIGGER *trigger name*

Table 1–2 (Cont.) Supported Teradata DDL

Operations	Object
RENAME	TABLE *table name* TO
	VIEW view name TO
	MACRO macro name
	TRIGGER *trigger name*
GRANT ... ON	TABLE *table name*
REVOKE ... ON	VIEW *view name*
	MACRO macro name
REPLACE	[RECURSIVE] VIEW *view name*
	TRIGGER *trigger name*
	MACRO macro name
COMMENT ON[3]	TABLE *table name*
	COLUMN *table name.column name*
	VIEW *view name*
	COLUMN *view name.column name*
	MACRO macro name
	TRIGGER *trigger name*
COLLECT STATISTICS ON[4]	*table name*
DROP STATISTICS ON	

[1] DDL operations on tables that are members of a replication group are automatically captured.

[2] DDL statements that refer to the temporary materialized state of the table cannot be replicated.

[3] Only COMMENT statements that create a user-defined description of an object in the data dictionary are captured.

[4] Only the optimizer form that is used by the Optimizer for generating table access and join plans is captured.

Note: The actual size limit of the DDL support is approximate, because the size will not only include 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.

The following DDL statements that change the properties of replication groups will be replicated automatically.

- ALTER REPLICATION GROUP with ADD and/or DROP clauses
- CREATE REPLICATION RULESET
- REPLACE REPLICATION RULESET
- DROP REPLICATION RULESET

Note: An ALTER REPLICATION GROUP statement that is used to generate a new security token will not be replicated.

For additional support limitations and configuration instructions, see *Chapter 5, "Configuring DDL Synchronization for a Teradata Database."*

2

Installing Oracle GoldenGate

This chapter describes installing Oracle GoldenGate for the first time and contains the following sections:

- Section 2.1, "Installation Overview"
- Section 2.2, "Downloading Oracle GoldenGate"
- Section 2.3, "Setting Library Paths for Dynamic Builds on UNIX"
- Section 2.4, "Preparing to Install Oracle GoldenGate within a Cluster"
- Section 2.5, "Installing Oracle GoldenGate on Linux and UNIX"
- Section 2.6, "Installing Oracle GoldenGate on Windows"
- Section 2.7, "Integrating Oracle GoldenGate into a Cluster"

2.1 Installation Overview

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/1212/gg-winux/docs.htm`

Oracle GoldenGate operates on a *replication server*, which is a Linux, UNIX, or Windows server that is separate from the servers that contain the Teradata databases. There can be one replication server, or there can be one for the source and another for the target. See Section 1.4, "Supported Platforms for a Replication Server" for additional information.

2.2 Downloading Oracle GoldenGate

Download the appropriate Oracle GoldenGate build to each system that will be part of the Oracle GoldenGate configuration.

1. Navigate to `http://edelivery.oracle.com`.

   The Oracle Software Delivery Cloud page appears.

2. Click **Sign-in/Register**.

> **Note:** If you are not already logged in, the Single Sign-on page appears. Enter your Oracle ID and password and click **Sign In**.

The Terms & Restrictions page appears

3. Accept the Oracle Software Delivery Cloud Trial License Agreement and the Export Restrictions and click **Continue**.

   The Media Pack Search page appears.

4. On the Media Pack Search page, do the following:

   a. Click the **Select Product Pack** drop-down control and, from the list, select Oracle Fusion Middleware.

   b. Click the **Platform** drop-down control and, from the list, select the platform on which you are installing Oracle GoldenGate.

   c. Click **Go**.

   The Results list expands to show all available media packs that include your search criteria.

5. In the Results list, select the media pack you want to download and click **Continue**.

   The media pack's download page appears. Multiple download selections may appear, such as separate builds for different databases or versions of a database. Note that this page contains the part number and size of each downloadable file.

6. To ensure that you download the files successfully, first review the *Media Pack Readme* for download instructions and product information by clicking **Readme**. The Readme contains release notes for any new features, new requirements, or bug fixes that affect your current configuration and other known issues.

7. To begin the download process, click **Download** next to the name of the Oracle GoldenGate build that you want to download.

   A File Download dialog box appears.

8. Select either **Open with** or **Save File**:

To...	Select...
Install media pack immediately	**Open**, select the desired file extraction utility, and extract the files to a designated location on your file system.
Save the file for later installation	**Save** and point to a designated location on your file system.

2.3 Setting Library Paths for Dynamic Builds on UNIX

Oracle GoldenGate uses shared libraries. When you install Oracle GoldenGate on a UNIX system, and you will be running the GGSCI program outside the Oracle GoldenGate installation directory, the following must be done *before* you run GGSCI or any other Oracle GoldenGate process:

- (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:

Table 2–1 Command Requiring Library Variable

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

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
```

To set the variables in Bourne shell:

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

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 2–2:

Table 2–2 UNIX/Linux Library Path Variables Per Platform

Platform[1]	**Environment variable**
IBM AIX IBM z/OS	LIBPATH
HP-UX	SHLIB_PATH
Sun Solaris HP Tru64 (OSF/1) LINUX	LD_LIBRARY_PATH[2]

[1] A specific platform may or may not be supported by Oracle GoldenGate for your database.

[2] In 64-bit environments with 32-bit Oracle databases, Oracle GoldenGate requires the `LD_LIBRARY_PATH` to include the 32-bit Oracle libraries.

Example

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

> **Note:** To view the libraries that are required by an 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.

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

You will need to install at least some Oracle GoldenGate objects on shared storage. Select cluster-aware shared storage that is independent of, but available to, all nodes of the cluster. The best practice is the install Oracle GoldenGate entirely on shared storage. This allows you to start the Oracle GoldenGate processes from any of the nodes without having to make changes to the parameter files. If the active node fails, the processes can be started quickly on another node, using the processing checkpoints that are preserved in the installation directory.

If you decide to install the Oracle GoldenGate binaries and files on each node, rather than on shared storage, the following must be true:

- 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.
  - `dirchk`
  - `dirdat`

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

- The parameter files in the `dirprm` directory, if not placed on the shared drive, must be identical on all nodes. To resolve environment settings that must be different from one node to the other, you can set environment settings so they are inherited from the local Manager process or reference a node-specific Oracle GoldenGate macro file. Because this scenario can be difficult to enforce, the inherent concerns can be avoided by storing the parameter files on the shared drive.

See also Section 2.7, "Integrating Oracle GoldenGate into a Cluster" after you install Oracle GoldenGate.

2.5 Installing Oracle GoldenGate on Linux and UNIX

Follow these steps to install Oracle GoldenGate for Oracle on a Linux or UNIX system or in the appropriate location in a cluster. See Section 2.4, "Preparing to Install Oracle GoldenGate within a Cluster" for more information.

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

   ```
   GGSCI
   ```

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

```
CREATE SUBDIRS
```

6. Issue the following command to exit GGSCI.

```
EXIT
```

7. Install the Teradata Access Module (TAM) library into the root Oracle GoldenGate directory on the replication server. The TAM communicates with an Oracle GoldenGate API that is known as a *Vendor Access Module*, or *VAM*. The VAM passes transactional data changes to the Extract process. For instructions on pairing the correct TAM version with your Teradata version, and for configuring the TAM for use with the Teradata database and Oracle GoldenGate, see the *Teradata Replication Services Using Oracle GoldenGate* documentation at `http://www.info.teradata.com`. In general, the TAM version should match the database version.

2.6 Installing Oracle GoldenGate on Windows

Follow these steps to install Oracle GoldenGate for Oracle on a Windows system or in the appropriate location in a cluster. See Section 2.4, "Preparing to Install Oracle GoldenGate within a Cluster" for more information.

Section 2.6.1, "Installing Oracle GoldenGate into a Windows Cluster"

Section 2.6.2, "Installing the Oracle GoldenGate Files"

Section 2.6.3, "Specifying a custom Manager name"

Section 2.6.4, "Installing Manager as a Windows Service"

2.6.1 Installing Oracle GoldenGate into a Windows Cluster

To install Oracle GoldenGate into a Windows cluster:

1. Log into one of the nodes in the Windows cluster.
2. Choose a drive for the Oracle GoldenGate installation location. This drive must be a resource within the same Windows cluster group that contains the database instance.
3. Ensure that this Windows cluster group is owned by the cluster node that you are logging into.
4. Install Oracle GoldenGate according to the following instructions.

2.6.2 Installing the Oracle GoldenGate Files

To install the Oracle GoldenGate files:

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

   `C:\"Oracle GoldenGate"` is not valid.

   `C:\Oracle_GoldenGate` is valid.
3. From the Oracle GoldenGate folder, run the GGSCI program.

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

```
CREATE SUBDIRS
```

5. Issue the following command to exit GGSCI.

```
EXIT
```

6. Install the Teradata Access Module (TAM) library into the root Oracle GoldenGate directory on the replication server. The TAM communicates with an Oracle GoldenGate API that is known as a *Vendor Access Module*, or *VAM*. The VAM passes transactional data changes to the Extract process. For instructions on pairing the correct TAM version with your Teradata version, and for configuring the TAM for use with the Teradata database and Oracle GoldenGate, see the *Teradata Replication Services Using Oracle GoldenGate* documentation at http://www.info.teradata.com. In general, the TAM version should match the database version.

2.6.3 Specifying a custom Manager name

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

- You want to use a name for Manager other than the default of GGSMGR.
- There will be multiple Manager processes running as Windows services on this system. Each Manager on a system must have a unique name. Before proceeding further, note the names of any local Manager services.

To specify a custom Manager name:

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

```
EDIT PARAMS ./GLOBALS
```

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

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

```
MGRSERVNAME name
```

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

2.6.4 Installing Manager as a Windows Service

By default, Manager is not installed as a service and can be run by a local or domain account. However, when run this way, Manager will stop when the user logs out. When you install Manager as a service, you can operate it independently of user connections, and you can configure it to start manually or at system start-up.

Installing Manager as a service is required on a Windows Cluster, but optional otherwise.

To install Manager as a Windows service:

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

   ```
   install option [...]
   ```

   Where: *option* is one of the following:

Table 2–3 INSTALL Utility Options

Option	Description
ADDEVENTS	Adds Oracle GoldenGate events to the Windows Event Manager.
ADDSERVICE	Adds Manager as a service with the name that is specified with the MGRSERVNAME parameter in the GLOBALS file, if one exists, or by the default of GGSMGR. ADDSERVICE configures the service to run as the Local System account, the standard for most Windows applications because the service can be run independently of user logins and password changes. To run Manager as a specific account, use the USER and PASSWORD options.[1] The service is installed to start at system boot time (see AUTOSTART). To start it after installation, either reboot the system or start the service manually from the Services applet of the Control Panel.
AUTOSTART	Sets the service that is created with ADDSERVICE to start at system boot time. This is the default unless MANUALSTART is used.
MANUALSTART	Sets the service that is created with ADDSERVICE to start manually through GGSCI, a script, or the Services applet of the Control Panel. The default is AUTOSTART .
USER *name*	Specifies a domain user account that executes Manager. For the *name*, include the domain name, a backward slash, and the user name, for example HEADQT\GGSMGR . By default, the Manager service is installed to use the Local System account.
PASSWORD *password*	Specifies the password for the user that is specified with USER .

[1] A user account can be changed by selecting the Properties action from the Services applet of the Windows Control Panel.

4. If Windows User Account Control (UAC) is enabled, you are prompted to allow or deny the program access to the computer. Select **Allow** to enable the INSTALL utility to run.

   The INSTALL utility installs the Manager service with a local system account running with administrator privileges. No further UAC prompts will be encountered when running Manager if installed as a service.

Note: If Manager is not installed as a service, Oracle GoldenGate users will receive a UAC prompt to confirm the elevation of privileges for Manager when it is started from the GGSCI command prompt. Running other Oracle GoldenGate programs also triggers a prompt.

2.7 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.

2.7.1 General Requirements in a Cluster

The general requirements for integrating Oracle GoldenGate into a cluster are:

1. Register the Oracle GoldenGate Manager process (and only Manager) as a cluster-managed resource as you would any other application. Manager must be the only Oracle GoldenGate process that the cluster-management software starts and stops, because it is the parent process that manages all other processes.
2. If the cluster uses a virtual IP address, you may need to obtain an available fixed IP address for the Manager process. The VIP must be an available IP address on the public subnet and cannot be determined through DHCP. In the parameter files of the Extract data pumps, specify the VIP of the remote Manager as the input value of the `RMTHOST` parameter. Other Oracle GoldenGate products that access Manager also should use the VIP.
3. When you configure Manager, add the `AUTOSTART` and `AUTORESTART` parameters so that Manager starts the replication processes automatically. You can, when needed, control Extract, Replicat, and other Oracle GoldenGate processes from within the Oracle GoldenGate user interfaces. For more information about these parameters, see *Reference for Oracle GoldenGate for Windows and UNIX*.
4. 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.
5. Configure Oracle GoldenGate as directed in this documentation.

2.7.2 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 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 (configurable 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 System for Oracle GoldenGate

This chapter contains guidelines for preparing the database and the system to support Oracle GoldenGate. This chapter contains the following sections:

- Section 3.1, "Preparing Tables for Processing"
- Section 3.2, "Creating a Teradata Replication Group"
- Section 3.3, "Activating DDL Capture by the Teradata TAM"
- Section 3.4, "Disabling the Capture of DDL"
- Section 3.5, "Configuring the TAM Initialization File"

3.1 Preparing Tables for Processing

The following table attributes must be addressed in an Oracle GoldenGate environment.

Section 3.1.1, "Disabling Triggers and Cascade Constraints"

Section 3.1.2, "Assigning Row Identifiers"

3.1.1 Disabling Triggers and Cascade Constraints

Disable triggers, cascade delete constraints, and cascade update constraints on target Sybase tables, or alter them to ignore changes made by the Oracle GoldenGate database user. Oracle GoldenGate replicates DML that results from a trigger or 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.

3.1.2 Assigning Row Identifiers

Oracle GoldenGate requires some form of unique row identifier on the source and target tables to locate the correct target rows for replicated updates and deletes.

3.1.2.1 How Oracle GoldenGate Determines the Kind of Row Identifier to Use

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:

1. Primary key
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
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 those that are not supported by Oracle GoldenGate in a key or those that are excluded from the Oracle GoldenGate configuration.

> **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.

3.1.2.2 Using KEYCOLS to Specify a Custom Key

If a table does not have one of the preceding types of row identifiers, or if you prefer those identifiers 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*.

3.2 Creating a Teradata Replication Group

To create a replication group, use a `CreateGroupStmtFile` (Create Group Statement) file. By using a `CreateGroupStmtFile`, the correct identifier information for the replication group is automatically written to the `tam.ini` file. For more information, see the *Teradata Replication Services Using Oracle GoldenGate* documentation.

All objects that have dependencies on one another must be specified in the same replication group. A transaction must be wholly contained within the same replication group.

To create a Create Group Statement file:

1. Use a text editor to create a text file.
2. Add the following lines:
   - The Teradata command `create replication group`
   - The name of the Teradata replication group

- The table list that is associated with the replication group. A table can only be associated with one replication group. Only one replication group is allowed per Oracle GoldenGate Extract group.

3. Save the file with the suffix `.sql` in a directory within the Oracle GoldenGate installation directory, either at the root level or in a sub-directory that you create, such as `dirtam`.
4. Specify the name of this file with the `CreateGroupStmtFile` parameter in the TAM initialization file. See Section 3.5, "Configuring the TAM Initialization File" for more information.

Example 3–1 Sample Create Group Statement File

```
Create Replication Group HRTRG1 (HR.EMPLOYEE, HR.EMP_INFO, HR.EMP_DEPT, HR.EMP_
REVIEWS);
```

3.3 Activating DDL Capture by the Teradata TAM

To specify DDL that you want the Teradata TAM to pass to Oracle GoldenGate, you create a replication ruleset statement for a replication group. A ruleset statement creates a set of one or more DDL-capture rules and associates them with the specified replication group. The rules are applied against the names and types of the target objects of the DDL statements as those operations are executed, making them immediately available for replication. DDL operations on tables that are members of a replication group are automatically captured.

> **Note:** The Teradata RSG must be properly configured. For more information, see the *Teradata Replication Services Using Oracle GoldenGate* documentation at `http://www.info.teradata.com`.

To activate DDL capture by the Teradata TAM:

1. Log in as a user with `REPLCONTROL` privilege.
2. Create the replication ruleset.

```
[CREATE | REPLACE] REPLICATION RULESET ruleset
[, DEFAULT]
FOR replication_group
AS rule_specification [, rule_specification]
```

Where:

- *rule_specification* is:

```
object_kind LIKE string_literal [ESCAPE character_literal]
[AND NOT LIKE string_literal [ESCAPE character_literal] ]
```

- *object_kind* is:

```
TABLE | TEMPORARY TABLE | VIEW | MACRO | TRIGGER | INDEX
```

Usage requirements

- A replication group must contain explicitly defined members with a table list, or it must be an empty group that is associated with a replication ruleset.
- If the `CREATE` form of the statement is used, and a rule set with the same rule set name already exists for the specified replication group, the `CREATE` statement fails.

- If the REPLACE form of the statement is used, and a rule set with the same rule set name already exists for the specified replication group, the existing ruleset is replaced by the new rule set.
- If the DEFAULT option is used, all of the rules in the rule set are considered to be default rules. A default rule is applied if no other rule matches the object.
  - A non-default rule must not match the same object as a non-default rule that is associated with another replication group.
  - A default rule must not match the same object as a default rule that is associated with another replication group.
  - The LIKE and NOT LIKE clauses specify pattern strings to match against the fully-qualified names of the objects of the SQL statements. The pattern strings can contain wildcard characters. The pattern and the optional ESCAPE character are used together in the same way as the LIKE predicate operator.

For more information about creating replication groups and rulesets, see the *Teradata Replication Services Using Oracle GoldenGate* documentation at http://www.info.teradata.com.

Example

This example creates a rule set named Sales1 for the replication group named MyRepGroup to capture any table that is created in the database named SalesDB and also capture any DDL that affects any view in SalesDB where the view name does not have the suffix _s.

```
CREATE REPLICATION GROUP MyRepGroup
CREATE REPLICATION RULESET Sales1 FOR MyRepGroup AS
TABLE LIKE 'SalesDB.%',
VIEW LIKE 'SalesDB.%' AND NOT LIKE '%z_s' ESCAPE 'z'
```

Note the use of an escape character to override the normal treatment of the underscore ("_") as a wildcard.

To configure Oracle GoldenGate DDL replication, see *Chapter 5, "Configuring DDL Synchronization for a Teradata Database."*

3.4 Disabling the Capture of DDL

This section contains instructions for disabling the capture and replication of Teradata DDL in either of two cases: Keeping the replication group or deleting the replication group.

Both cases require REPLCONTROL privilege.

To disable the capture of DDL and keep the replication group:

```
DROP REPLICATION RULESET ruleset FOR replication_group
```

For example:

```
DROP REPLICATION RULESET Sales1 FOR MyRepGroup;
```

To disable the capture of DDL and delete the replication group:

```
DROP REPLICATION GROUP replication_group [ ( table [, ... ] )]
```

For example:

```
DROP REPLICATION GROUP MyRepGroup
```

3.5 Configuring the TAM Initialization File

The Teradata Access Module (TAM) provides the Extract process with links to the Teradata environment. To configure the TAM, create an initialization file.

The following is an example TAM initialization file.

```
Mode=Replication
DictOdbcConnString=DSN=myDsn;uid=myUser;pwd=myPass
MgmtOdbcConnString=DSN=myDsn;uid=myUser;pwd=myPass
CreateGroupStmtFile=c:\GGS\Teradata\dirtam\hrtrg1.sql
CharacterSet=ASCII
ControlRSG=10.10.10.49:1152
DataRSG1=10.10.10.50:1153
DataRSG2=node3
DataRSG3=node4:1155
Encryption=All
```

Table 3–1 describes the required TAM initialization parameters.

1. Use a text editor to create a text file.
2. Add the following required parameters to the file. Optional parameters that are listed also can be used as needed.

Table 3–1 Required TAM initialization File Parameters

Parameter	Description
`Mode={Replication}`	*Required* Specifies that the replication mode will be change data capture.
`DictOdbcConnString=`*`ODBC_connection_string_for_metadata`*	*Required* The logon string of a user with access rights to the dictionary tables.
`MgmtOdbcConnString=`*`ODBC_connection_string for_management_functions`*	*Required* The logon string of a user with rights to execute management functions, such as `CREATE REPLICATION GROUP`. This logon requires the `REPLCONTROL` privilege.
`CreateGroupStmtFile=` *`name`*	*Required for TAM12 and later* The name of the Create Group statement file that contains the `CREATE REPLICATION GROUP` statement for a new group. If the replication group was not created with a Create Group statement file, omit or comment out this parameter and use the `GroupID` and `SecurityToken` parameters.

`GroupID=`
`ID`
	Required if SecurityToken is used
	The ID of the replication group that is associated with the TAM. If `GroupID` is used, `SecurityToken` must be used.
	You can view the ID of any replication group with this command:
	`HELP REPLICATION GROUP` *`replication_group`*`;`
	For example:
	`help replication group g1;`
	The ID is a numerical ID in the `Identifier` column next to the name of the group.

Table 3–1 (Cont.) Required TAM initialization File Parameters

Parameter	Description
`SecurityToken=`*`token`*	*Required if replication group was not created with Create Group Statement* The security token for the replication group that is associated with the TAM. If a group was created with a Create Group Statement file and you specify that file with the `CreateGroupStmtFile` parameter, the `SecurityToken` and `GroupID` parameters can be omitted because they will be generated automatically at runtime.
`AltControlRSG=`*`IP_or_name`*	*Optional* Specifies the IP address or name of a server that can take over as the control RSG when the primary one fails.
`CharacterSet=` `{ASCII | UTF16}`	*Required* The character set for this replication group. There is more overhead associated with UTF16, so use it only when required

`ControlRSG=`*`address`*`[:`*`port`*`]`
`DataRSG1=`*`address`*`[:`*`port`*`]`
`[DataRSG2=`*`address`*`[:`*`port`*`]`
`] [...]`
	Required
	The RSG node addresses. Can be either a node name or an IP address and can be appended with an optional port number. The control RSG should be the highest numbered of the RSGs in the system.

`Encryption=`
`{None | Control | Data |`
`All}`
	Required
	The types of messages to encrypt. There is little difference in overhead between encrypting just data messages and encrypting both data and control messages. Use `All` to specify encryption or `None` to specify no encryption.

`RsgTimeoutSec=`*`seconds`*	*Optional* The timeout, in seconds, when polling RSG for data. The default is 1. Valid range is 0-60 seconds.
`RsgTimeoutMSec=`*`milliseconds`*	*Optional* The timeout, in milliseconds, when polling RSG for data. The default is 0. Valid range is 0-60000000 milliseconds.

`Tracing= {`
`Debug |`
`Performance |`
`All |`
`None}`
	Optional
	The level of debug tracing.
	The default is `None`.

`MaxProtTransCompleteThresh=`*`number_of_transactions`*	*Optional, valid for max protection mode* The number of outstanding transactions that can be held after which an Oracle GoldenGate checkpoint must be requested. This is significant when operating in maximum protection mode. If there are numerous sessions that are applying transactions at a high rate, this parameter can be set to a higher value, such as 10. However, if the number of sessions is small, or if the rate of submission is low, you can set it to a lower value, such as 1-4, to minimize latency and maximize throughput. The default is 0. Valid range is 0-24 transactions.
`Bidirectional` `{TRUE | FALSE}`	*Optional* Specifies whether or not before images of data are sent to Oracle GoldenGate. Must be `TRUE` if the Extract parameter `GETUPDATEBEFORES` is used. The default is `FALSE`, which sends only the after image of data to reduce the CDC overhead and communication bandwidth that is used.

3. Save the file as an ASCII file named `tam.ini` within the Oracle GoldenGate installation directory, either at the root level or in a sub-directory that you create, such as `dirtam`.
4. Specify the name of this file with the `VAM` parameter in the Extract parameter file.

4

Configuring Oracle GoldenGate

This chapter describes how Oracle GoldenGate works with the Change Data Capture (CDC) component of a source Teradata database to operate in either *maximum protection mode* or *maximum performance mode*. The mode determines the commit protocol that is used and whether or not Oracle GoldenGate has an effect on the Teradata applications. This chapter contains the following sections:

- Section 4.1, "Configuring Oracle GoldenGate in Maximum Protection Mode"
- Section 4.2, "Configuring Oracle GoldenGate in Maximum Performance Mode"
- Section 4.3, "Additional Oracle GoldenGate Configuration Guidelines"

4.1 Configuring Oracle GoldenGate in Maximum Protection Mode

Maximum protection mode is the recommended Oracle GoldenGate configuration for the Teradata database. Maximum protection mode incorporates Oracle GoldenGate into the production system by using a two-phase commit protocol between CDC on the source server and a primary Extract process on the replication server (through the TAM). The two-phase commit requires a source transaction to be committed to Oracle GoldenGate as well as to the source database, ensuring that no transactions are lost in transit or duplicated if communication is interrupted or a component fails.

In this mode, a transaction is *in doubt* (not committed) until the primary Extract acknowledges that it received all of the data and saved it to an Oracle GoldenGate VAM trail on disk.

- If CDC receives the acknowledgement within a given timeout period, it releases the transaction for commit to the application, for commit to the database, and for propagation by Oracle GoldenGate.
- If CDC does not receive the acknowledgement within a given timeout period, it rolls back the transaction, and the application user receives an error message.

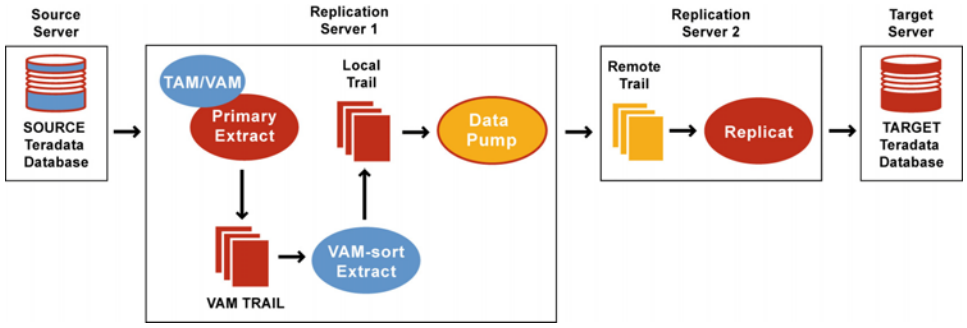
The VAM trail is a series of files that work like a transaction log. It stores incoming data in the order that it is received, but not necessarily in transaction order. A secondary Extract process, known as a *VAM-sort Extract*, sorts the data into transaction order and either deletes a transaction if a rollback is received (because the two-phase commit failed) or releases it to a regular trail for further processing.

4.1.1 Recommended Maximum Protection Configuration

Install the Extract and Replicat processes on separate replication servers and use a data pump with a local trail on the server where the Extract processes are installed. In this configuration, the primary Extract group captures the data, and then the VAM-sort Extract persists the sorted data to a regular Oracle GoldenGate trail on the local disk.

A data-pump Extract reads this trail and sends the data across TCP/IP to a trail on the Replicat replication server, where it is read again by a Replicat process and applied to the target. If there is a failure of communication between the Extract server and the Replicat server, only the data pump is affected. The other two Extract processes can continue to do their work without running out of memory if the outage persists.

Figure 4–1 Recommended Maximum Protection Configuration



This graphic shows how maximum protection mode persists transaction data to a VAM-sort trail that is read by a VAM-sort Extract. The VAM-sort Extract sends only committed transactions, in the correct order, to a local trail that is read by a data pump for transmission to the target.

****

4.1.2 Configuring Extract in Maximum Protection Mode

Perform these steps on the source replication server.

1. Configure the Manager process according to the instructions in Administering Oracle GoldenGate for Windows and UNIX.
2. In the Manager parameter file, use the `PURGEOLDEXTRACTS` parameter to control the purging of files from the local trail.
3. Run GGSCI.
4. Create a primary Extract group. For documentation purposes, this group is called `ext`.

   ```
   ADD EXTRACT ext, VAM
   ```

5. Create a local trail that is to be the VAM trail.

   ```
   ADD EXTTRAIL VAM_trail, EXTRACT ext
   ```

   Use the `EXTRACT` argument to link this trail to the primary Extract group. That Extract group creates this trail as a VAM trail.

6. Use the `EDIT PARAMS` command to create a parameter file for the primary Extract group. Include the parameters shown in Example 4–1 plus any others that apply to your database environment.

> **Note:** Encryption options exist for encrypting the database password, the data in the trail, and the data sent across TCP/IP. See *Administering Oracle GoldenGate for Windows and UNIX* for more information about security features.

Example 4–1 Parameters for the Primary Extract Group

```
-- Identify the Extract group:
EXTRACT ext
-- Specify database login information as needed for the database:
[SOURCEDB dsn1,][USERID user[, PASSWORD pw]]
-- Specify the VAM trail:
EXTTRAIL VAM_trail
-- Specify that this Extract creates and writes to a VAM trail:
DSOPTIONS CREATETRANLOG RESTARTAPPEND, VAMCOMPATIBILITY 1
-- Specify library, TAM initialization file, and callback program for the TAM:
VAM library, PARAMS (inifile, ini_file, callbackLib, extract.exe)
-- Specify tables to be captured:
TABLE owner.table;
```

7. Create a VAM-sort Extract group to read the VAM trail. For documentation purposes, this group is called `extsort`.

   ```
   ADD EXTRACT extsort, VAMTRAILSOURCE VAM_trail
   ```

8. Add a local trail to receive the sorted data.

   ```
   ADD EXTTRAIL local_trail, EXTRACT extsort
   ```

   Use the `EXTRACT` argument to link this trail to the VAM-sort Extract group. A data pump group will read this trail.

9. Use the `EDIT PARAMS` command to create a parameter file for the VAM-sort Extract group. Include the parameters shown in Example 4–2 plus any others that apply to your database environment.

Example 4–2 Parameters for the VAM-sort Extract Group

```
-- Identify the Extract group:
EXTRACT extsort
-- Specify database login information as needed for the database:
[SOURCEDB dsn1,][USERID user[, PASSWORD pw]]
-- Specify that this Extract reads a VAM trail and sorts the data:
DSOPTIONS SORTTRANLOG
-- Specify the local trail to receive the sorted data:
EXTTRAIL local_trail
-- Specify tables to be captured:
TABLE owner.table;
```

10. Create a data pump group to read the local trail and send the data to a remote trail on one of the following:

    - The replication server where Replicat is running against a target Teradata database.
    - A target server where Replicat is running against another database platform that is supported by Oracle GoldenGate.

    ```
    ADD EXTRACT pump, EXTTRAILSOURCE local_trail
    ```

    For documentation purposes, this group is called `pump`.

11. Add the remote trail.

    ```
    ADD RMTTRAIL remote_trail, EXTRACT pump
    ```

    Use the `EXTRACT` argument to link the remote trail to the data pump group.

12. Create a parameter file for the data pump. Include the parameters shown in Example 4–3 plus any others that apply to your database environment.

Example 4–3 Parameters for Data Pump Extract Group

```
-- Identify the data pump group:
EXTRACT pump
-- Specify database login information as needed for the database:
[SOURCEDB dsn1,][USERID user[, PASSWORD pw]]
-- Specify the name or IP address of the remote system:
RMTHOST target, MGRPORT portnumber
-- Specify the remote trail:
RMTTRAIL remote_trail
-- Allow mapping, filtering, conversion or pass data through as-is:
[PASSTHRU | NOPASSTHRU]
-- Specify tables to be captured:
TABLE owner.table;
```

Note: To use `PASSTHRU` mode, the names of the source and target objects must be identical. No column mapping, filtering, `SQLEXEC` functions, transformation, or other functions that require data manipulation can be specified in the parameter file. You can combine normal processing with pass-through processing by pairing `PASSTHRU` and `NOPASSTHRU` with different `TABLE` statements.

4.1.3 Configuring Replicat to Support Maximum Protection Mode

Replicat does not require any special configuration with regard to maximum protection mode. This section highlights basic Replicat parameters that are required for most target database types. Additional parameters may be required. See the Oracle GoldenGate installation and configuration documentation for your target database and also *Reference for Oracle GoldenGate for Windows and UNIX* reference documentation.

Perform these steps on the target replication server or target database system.

1. Configure the Manager process according to the instructions in Administering Oracle GoldenGate for Windows and UNIX.
2. In the Manager parameter file, use the `PURGEOLDEXTRACTS` parameter to control the purging of files from the local trail.
3. Create a Replicat checkpoint table. There are multiple options for this purpose. For instructions, see *Administering Oracle GoldenGate for Windows and UNIX*.
4. Create a Replicat group. For documentation purposes, this group is called `rep`.

   ```
   ADD REPLICAT rep, EXTTRAIL remote_trail
   ```

   Use the `EXTTRAIL` argument to link the Replicat group to the remote trail that you specified for the data pump on the source server.
5. Use the `EDIT PARAMS` command to create a parameter file for the Replicat group. Include the parameters shown in Example 4–4 plus any others that apply to your database environment.

Example 4–4 Parameters for the Replicat Group

```
-- Identify the Replicat group:
REPLICAT rep
```

```
-- State whether or not source and target definitions are identical:
SOURCEDEFS {full_pathname | ASSUMETARGETDEFS}
-- Specify database login information as needed for the database:
[TARGETDB dsn2,] [USERID user id[, PASSWORD pw]]
-- Specify error handling rules (See the NOTE following parameter file):
REPERROR (error, response)
-- Specify tables for delivery:
MAP owner.table, TARGET owner.table[, DEF template name];
```

Note: Teradata Multiload does not participate in the full two-phase commit protocol of maximum protection mode. In a recovery situation, it is possible that Replicat could attempt to apply some updates twice. If a multiset table is affected, this could result in duplicate rows being created. Use the REPERROR parameter in the Replicat parameter file so that Replicat ignores duplicate rows.

4.2 Configuring Oracle GoldenGate in Maximum Performance Mode

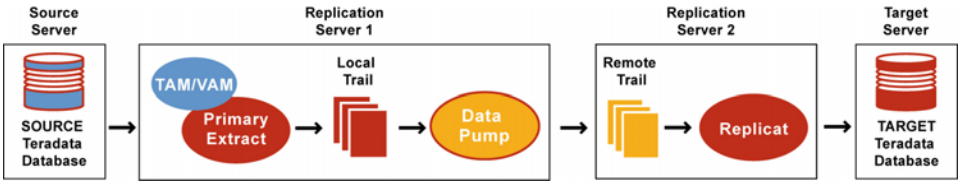
Maximum performance mode is faster and less intrusive than maximum protection mode, but it is less fault tolerant. When the source application issues a commit, CDC begins transmitting the data to the replication server, where it is buffered and sorted by Extract. When the data is finished being transmitted, CDC sends Extract a commit and releases the transaction for commit to the application and to the database.

The maximum performance configuration does not persist incoming data to disk, nor does it have an acknowledgement system between CDC and Extract that prevents data loss. If communication between the primary Extract and the Teradata source is interrupted, or if a component fails, the source and target data are no longer synchronized.

4.2.1 Recommended Maximum Performance Configuration

Install the Extract and Replicat processes on separate replication servers, and use a data pump with a local trail on the Extract server. In this configuration, the primary Extract persists transactions to a local Oracle GoldenGate trail. A data pump Extract reads this trail and sends the data across TCP/IP to a trail on the Replicat server, where it is read again by Replicat and applied to the target. If there is a failure of communication between the Extract and Replicat replication servers, only the data pump is affected. The primary Extract can continue to write incoming data to disk instead of having to retain it in memory, which otherwise could be depleted in a prolonged outage.

Figure 4–2 Recommended Maximum Performance Configuration



This graphic shows how Extract writes transaction data directly to a local trail, and a data pump Extract reads the trail and sends the data across TCP/IP to the target system.

```
**********************************************************************************************
```

4.2.2 Configuring Extract in Maximum Performance Mode

Perform these steps on the source replication server.

1. Configure the Manager process according to the instructions in *Administering Oracle GoldenGate for Windows and UNIX*.
2. In the Manager parameter file, use the `PURGEOLDEXTRACTS` parameter to control the purging of files from the local trail.
3. Run GGSCI.
4. Create a primary Extract group. For documentation purposes, this group is called `ext`.

   ```
   ADD EXTRACT ext, VAM
   ```

5. Add a local trail.

   ```
   ADD EXTTRAIL local_trail, EXTRACT ext
   ```

   Use the `EXTRACT` argument to link this trail to the primary Extract group.
6. Use the `EDIT PARAMS` command to create a parameter file for the primary Extract group. Include the parameters in Example 4–5 plus any others that apply to your database environment.

   Note: Encryption options exist for encrypting the database password, the data in the trail, and the data sent across TCP/IP. See *Administering Oracle GoldenGate for Windows and UNIX* for more information about security features.

Example 4–5 Parameters for the Primary Extract Group

```
EXTRACT ext
-- Identify the Extract group:
-- Specify database login information as needed for the database:
[SOURCEDB dsn1,][USERID user[, PASSWORD pw]]
-- Specify the local trail that this Extract writes to:
EXTTRAIL local_trail
-- Specify that this Extract is in maximum performance mode:
DSOPTIONS COMMITTEDTRANLOG RESTARTAPPEND, VAMCOMPATIBILITY 1
-- Specify library, TAM initialization file, and callback program for the TAM:
VAM library, PARAMS (inifile, ini_file, callbackLib, extract.exe)
-- Specify tables to be captured:
TABLE owner.table;
```

7. Create a data pump group to read the local trail and send the data to a remote trail on one of the following:
   - The replication server where Replicat is running against a target Teradata database.
   - A target server where Replicat is running against another database platform that is supported by Oracle GoldenGate.

   ```
   ADD EXTRACT pump, EXTTRAILSOURCE local_trail
   ```

   For documentation purposes, this group is called `pump`.

8. Add the remote trail.

   ```
   ADD RMTTRAIL remote_trail, EXTRACT pump
   ```

   Use the EXTRACT argument to link the remote trail to the data pump group.

9. Create a parameter file for the data pump. Include the parameters shown in Example 4–6 plus any others that apply to your database environment.

Example 4–6 Parameters for the Extract Data Pump

```
-- Identify the data pump group:
EXTRACT pump
-- Specify database login information as needed for the database:
[SOURCEDB dsn1,][USERID user[, PASSWORD pw]]
-- Specify the name or IP address of the remote system:
RMTHOST target, MGRPORT portnumber
-- Specify the remote trail:
RMTTRAIL remote_trail
-- Allow mapping, filtering, conversion or pass data through as-is:
[PASSTHRU | NOPASSTHRU]
-- Specify tables to be captured:
TABLE owner.table
```

Note: To use PASSTHRU mode, the names of the source and target objects must be identical. No column mapping, filtering, SQLEXEC functions, transformation, or other functions that require data manipulation can be specified in the parameter file. You can combine normal processing with pass-through processing by pairing PASSTHRU and NOPASSTHRU with different TABLE statements.

4.2.3 Configuring Replicat to Support Maximum Performance Mode

Replicat does not require any special configuration with regard to maximum protection mode. This section highlights basic Replicat parameters that are required for most target database types. Additional parameters may be required. See the Oracle GoldenGate documentation for your target database and also *Reference for Oracle GoldenGate for Windows and UNIX*.

Perform these steps on the target replication server or target database system.

1. Configure the Manager process according to the instructions in *Administering Oracle GoldenGate for Windows and UNIX*.
2. In the Manager parameter file, use the PURGEOLDEXTRACTS parameter to control the purging of files from the local trail.
3. Create a Replicat checkpoint table. There are multiple options for this purpose. For instructions, see *Administering Oracle GoldenGate for Windows and UNIX*
4. Create a Replicat group. For documentation purposes, this group is called rep.

   ```
   ADD REPLICAT rep, EXTTRAIL remote_trail
   ```

   Use the EXTTRAIL argument to link the Replicat group to the remote trail that you specified for the data pump on the source server.

5. Use the EDIT PARAMS command to create a parameter file for the Replicat group. Include the parameters shown in Example 4–7 plus any others that apply to your database environment.

Example 4–7 Parameters for the Replicat Group

```
-- Identify the Replicat group:
REPLICAT rep
-- State whether or not source and target definitions are identical:
SOURCEDEFS full_pathname | ASSUMETARGETDEFS
-- Specify database login information as needed for the database:
[TARGETDB dsn2,] [USERID user id[, PASSWORD pw]]
-- Specify error handling rules:
REPERROR (error, response)
-- Specify tables for delivery:
MAP owner.table, TARGET owner.table[, DEF template name];
;
```

4.3 Additional Oracle GoldenGate Configuration Guidelines

The following are additional considerations to make once you have installed and configured your Oracle GoldenGate environment.

4.3.1 Configuring DDL Support

Oracle GoldenGate supports the replication of Teradata DDL. For more information, see "Configuring DDL Synchronization for a Teradata Database" on page 5-1.

4.3.2 Handling Massive Update and Delete Operations

Operations that update or delete a large number of rows will generate discrete updates and deletes for each row on the subscriber database. This could cause a lock manager overflow on the Teradata subscriber system, and thus terminate the Replicat process.

To avoid these errors, you can do either of the following:

- Temporarily suspend replication for these operations and then perform them manually on the source and target systems. To suspend replication, use the following command, which suspends replication for that session only. The operations of other sessions on that table are replicated normally.

  ```
  set session override replication on;
  commit;
  ```

- Set the Replicat parameter `MAXTRANSOPS` to a value of less than 1000. This parameter splits large transactions into smaller ones.

4.3.3 Preventing Multiple Connections

By default, the Extract and Replicat processes create a new connection for catalog queries. You can prevent this extra connection by using the `DBOPTIONS` parameter with the `NOCATALOGCONNECT` option.

4.3.4 Performing Initial Synchronization

Perform an initial synchronization of the source and target data before using Oracle GoldenGate to transmit transactional changes for the first time. The preferred methods for synchronizing two Teradata databases is to use any of the Teradata data loader utilities. The recommended utility is MultiLoad. To configure an initial load, see *Administering Oracle GoldenGate for Windows and UNIX*.

5

Configuring DDL Synchronization for a Teradata Database

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

- Section 5.1, "About this Chapter"
- Section 5.2, "Overview of DDL Synchronization"
- Section 5.3, "Limitations of Oracle GoldenGate DDL Support"
- Section 5.4, "Configuration Guidelines for DDL Support"
- Section 5.5, "Understanding DDL Scopes"
- Section 5.6, "Enabling DDL Support"
- Section 5.7, "Filtering DDL Replication"
- Section 5.8, "How Oracle GoldenGate Handles Derived Object Names"
- Section 5.9, "Using DDL String Substitution"
- Section 5.10, "How DDL is Evaluated for Processing"
- Section 5.11, "Handling DDL Processing Errors"
- Section 5.12, "Viewing DDL Report Information"
- Section 5.13, "Tracing DDL Processing"

5.1 About this Chapter

This chapter contains information that is specific to the setup of the Oracle GoldenGate solution within a Teradata environment. It assumes that the reader has a fundamental knowledge of the Teradata database and the Teradata Replication Solutions. It also assumes that the following have been configured properly:

- Relay Services Gateway (RSG)
- Change Data Capture (CDC)
- Teradata Access Module (TAM)
- Replication groups

For a complete description of how to configure replication for the Teradata database, see the Teradata Replication Solutions documentation.

5.2 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

5.3 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 Oracle GoldenGate release notes or the readme file that comes with the software.

5.3.1 DDL Statement Length

Oracle GoldenGate measures the length of a DDL statement in bytes, not in characters. The supported length is approximately 2 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.

5.3.2 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 uni-directional configurations, and in bi-directional configurations between two, and only two, systems.

5.3.3 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 or a VAM-sort Extract is not permitted, and the DDL is passed as it was received from the primary Extract. This is known as `PASSTHRU` mode.

For example, `ALTER TABLE TableA` is processed by a data pump or a VAM-sort Extract as `ALTER TABLE TableA`. It cannot be mapped by that process as `ALTER TABLE TableB`, regardless of any `TABLE` statements that specify otherwise.

5.3.4 DDL Response Time

The response time for DDL statements that are captured for replication might increase because of the inherent latency of the synchronization protocol between the Teradata database and the replication system, including the Oracle GoldenGate component. The response time overhead should not exceed one second under most conditions. The response time of DDL that is not captured should not be significantly affected. The performance cost of capturing changed data in tables with UDTs or LOBs, compared to tables without those data types, should be comparable to other methods of exporting data.

5.4 Configuration Guidelines for DDL Support

The following are guidelines to take into account when configuring Oracle GoldenGate processes to support DDL replication.

5.4.1 Database Privileges

See "Database User for Oracle GoldenGate Processes" on page 1-5 for database privileges that are required for Oracle GoldenGate to support DDL capture and replication.

5.4.2 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.

5.4.3 DDL and DML in Data Pumps and VAM-sort Extracts

If using a data pump or a VAM-sort Extract, configure DML for `PASSTHRU` mode if the objects are using DDL support. DDL is passed through a data pump or a VAM-sort Extract in `PASSTHRU` mode, so the same must be true of the DML. Any filtering, mapping, or transformation of the DML must be done by the primary Extract or by Replicat. However, tables that do not use DDL support can be configured in `NOPASSTHRU` mode to allow data filtering, and manipulation by a data pump.

To configure tables for PASSTHRU, NOPASSTHRU, or both

1. In the parameter file of the data pump or VAM-sort Extract, place the `PASSTHRU` parameter before all of the `TABLE` statements that contain tables that use DDL support.

2. In the same parameter file, you can place the NOPASSTHRU parameter before any TABLE statements that contain tables that do not use DDL support, if you want data filtering, mapping, or transformation to be performed for them.
3. Do not use any of the DDL configuration parameters for a data pump or VAM-sort Extract: DDL, DDLOPTIONS, DDLSUBST, DDLERROR, or any of the Oracle GoldenGate tracing parameters with DDL options.

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

5.4.4 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. For more information about support for object names, see the Oracle GoldenGate *Windows and UNIX Administrator's Guide*.

You can use the question mark (?) and asterisk (*) wildcards to specify object names in configuration parameters that support DDL synchronization. For more information about support for wildcards, see the Oracle GoldenGate *Windows and UNIX Administrator's Guide*. 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.

5.4.5 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 the Oracle GoldenGate *Windows and UNIX Reference Guide*.

5.4.6 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.

DDL support is also compatible with the Teradata FastLoad initial synchronization method. See the Teradata documentation for more information about this feature.

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.

5.5 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.

5.5.1 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 Table 5–1 are supported.

Table 5–1 Objects That Can Be Mapped in MAP and TABLE Statements

Operations	Object
CREATE	TABLE (includes AS SELECT)
ALTER	INDEX
DROP	TRIGGER
RENAME	VIEW
COMMENT ON	FUNCTION
	PROCEDURE
	MACRO
GRANT	TABLE
REVOKE	

The following limitations apply to MAPPED scope:

- TABLE and MAP do not support some special characters that could be used in an object name. For a list of those characters, see *Administering Oracle GoldenGate for Windows and UNIX*. Objects with non-supported special characters are supported by the scopes of UNMAPPED and OTHER.
- Support for COMMENT ON applies to COMMENT ON TABLE and COMMENT ON COLUMN.
- DDL on an index is not supported for HASH and JOIN operations.

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 owner and name in the TARGET clause of the MAP statement.

Assume the following TABLE and MAP statements and source DDL statement:

Extract (source)	Replicat (target)	Source DDL
TABLE fin.expen;	MAP fin.expen, TARGET fin2.expen2;	ALTER TABLE fin.expen ADD notes varchar2(100);
TABLE hr.tab*;	MAP hr.tab*, TARGET hrBackup.bak_*;	

In this example, because the source table fin.expen is in a MAP statement with a TARGET clause that maps to a different owner 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.

5.5.1.1 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 owner. Consider following ALTER INDEX...RENAME:

```
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;
```

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 becomes:

```
ALTER INDEX tgt.indnew RENAME TO indnew;
```

5.5.2 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 (that is, it is not contained in an Extract TABLE statement), but of MAPPED scope on the target (that is, it is contained in a Replicat MAP statement), or the other way around. When Teradata DDL is of UNMAPPED scope in the Replicat configuration, it is applied to the target in one of these ways:

- If the required Replicat connection parameter TARGETDB contains just a DSN (as in tdtarg), but not a database name, it is applied to the target object with the same owner (database name) and object name as in the source DDL.
- If a specific database name is used in TARGETDB (as in db@tdtarg), all of the DDL operations are applied to the target with the owner from TARGETDB.

5.5.3 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 owner 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.

5.6 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, although the Teradata TAM sends all of the DDL to the Oracle GoldenGate VAM. 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.

5.7 Filtering DDL Replication

Use the DDL parameter to filter DDL operations so that specific (or all) DDL is applied to the target database according to your requirements.

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 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 filtering criteria, it is recommended that you test your configuration in a test environment before using it in production.

> **Note:** Before you create a DDL parameter statement, it might help to review Section 5.10, "How DDL is Evaluated for Processing".

The following is the syntax for the DDL parameter.

```
DDL [
{INCLUDE | EXCLUDE}
```

```
[, MAPPED | UNMAPPED | OTHER | ALL]
[, OPTYPE type]
[, OBJTYPE 'type']
[, OBJNAME name]
[, INSTR 'string']
[, INSTRWORDS 'word_list']
[, EVENTACTIONS (action)
]
[...]
```

Table 5–2 DDL Inclusion and Exclusion Options

Option	Description

INCLUDE | EXCLUDE	Use INCLUDE and EXCLUDE to identify the beginning of an inclusion or exclusion clause.
	■ An inclusion clause contains filtering criteria that identifies the DDL that this parameter will affect.
	■ An exclusion clause contains filtering criteria that excludes specific DDL from this parameter.
	The inclusion or exclusion clause must consist of the INCLUDE or EXCLUDE keyword followed by any valid combination of other options of the parameter that is being applied.
	If you use EXCLUDE, you must create a corresponding INCLUDE clause. For example, the following is invalid:
	`DDL EXCLUDE OBJNAME hr.*`
	However, you can use either of the following:
	`DDL INCLUDE ALL, EXCLUDE OBJNAME hr.*`
	`DDL INCLUDE OBJNAME fin.* EXCLUDE fin.ss`
	An EXCLUDE takes priority over any INCLUDEs that contain the same criteria. You can use multiple inclusion and exclusion clauses.

MAPPED | UNMAPPED | OTHER | ALL	Use MAPPED, UNMAPPED, OTHER, and ALL to apply INCLUDE or EXCLUDE based on the DDL operation scope. ■ MAPPED applies INCLUDE or EXCLUDE to DDL operations that are of MAPPED scope. MAPPED filtering is performed before filtering that is specified with other DDL parameter options. ■ UNMAPPED applies INCLUDE or EXCLUDE to DDL operations that are of UNMAPPED scope. ■ OTHER applies INCLUDE or EXCLUDE to DDL operations that are of OTHER scope. ■ ALL applies INCLUDE or EXCLUDE to DDL operations of all scopes.
OPTYPE *type*	Use OPTYPE to apply INCLUDE or EXCLUDE to a specific type of DDL operation, such as CREATE, ALTER, and RENAME. For *type*, use any DDL command that is valid for the database. For example, to include ALTER operations, the correct syntax is: `DDL INCLUDE OPTYPE ALTER`
OBJTYPE '*type*'	Use OBJTYPE to apply INCLUDE or EXCLUDE to a specific type of database object. For *type*, use any object type that is valid for the database, such as TABLE, INDEX, and TRIGGER. Enclose the name of the object type within single quotes. For example: `DDL INCLUDE OBJTYPE 'INDEX'`

Table 5–2 (Cont.) DDL Inclusion and Exclusion Options

Option	Description

OBJNAME *name*	Use OBJNAME to apply INCLUDE or EXCLUDE to the fully qualified name of an object, for example *owner.table_name*. You can use a wildcard only for the object name.
	
	Example:
	
	`DDL INCLUDE OBJNAME accounts.*`
	
	When using OBJNAME with MAPPED in a Replicat parameter file, the value for OBJNAME must refer to the name specified with the TARGET clause of the MAP statement. For example, given the following MAP statement, the correct value is OBJNAME fin2.*.
	
	`MAP fin.exp_*, TARGET fin2.*;`
	
	In the following example, a CREATE TABLE statement executes like this on the source:
	
	`CREATE TABLE fin.exp_phone;`
	
	And like this on the target:
	
	`CREATE TABLE fin2.exp_phone;`
	
	If a target owner is not specified in the MAP statement, Replicat maps it to the database user that is specified with the USERID parameter.
	
	For DDL that creates derived objects, such as a trigger, the value for OBJNAME must be the name of the base object, not the name of the derived object.
	
	For example, to include the following DDL statement, the correct value is hr.accounts, not hr.insert_trig.
	
	`CREATE TRIGGER hr.insert_trig ON hr.accounts;`
	
	For RENAME operations, the value for OBJNAME must be the new table name. For example, to include the following DDL statement, the correct value is hr.acct.
	
	`ALTER TABLE hr.accounts RENAME TO acct;`

INSTR '*string*'	Use INSTR to apply INCLUDE or EXCLUDE to DDL statements that contain a specific character string within the command syntax. For example, the following excludes DDL that creates an index. `DDL INCLUDE ALL EXCLUDE INSTR 'CREATE INDEX'` Enclose the string within single quotes. The string search is not case sensitive. INSTR does not support single quotation marks (' ') that are within the string, nor does it support NULL values.

Table 5–2 (Cont.) DDL Inclusion and Exclusion Options

Option	Description

INSTRWORDS '*word_list*'	Use INSTRWORDS to apply INCLUDE or EXCLUDE to DDL statements that contain the specified words.
	For *word_list*, supply the words in any order, within single quotes. To include spaces, put the space (and the word, if applicable) in double quotes. Double quotes also can be used to enclose sentences.
	All specified words must be present in the DDL for INSTRWORDS to take effect.
	Example:
	`ALTER TABLE INCLUDE INSTRWORDS 'ALTER CONSTRAINT " xyz"`
	This example will match this:
	`ALTER TABLE ADD CONSTRAINT xyz CHECK`
	and this:
	`ALTER TABLE DROP CONSTRAINT xyz`
	INSTRWORDS does not support single quotation marks (' ') that are within the string, nor does it support NULL values.

EVENTACTIONS (*action*)	Causes the Extract or Replicat process take a defined action based on a DDL record in the transaction log or trail, which is known as the *event record*. The DDL event is triggered if the DDL record is eligible to be written to the trail by Extract or a data pump, or to be executed by Replicat, as determined by the other filtering options of the DDL parameter. You can use this system to customize processing based on database events. For *action* see EVENTACTIONS under the MAP and TABLE parameters. Guidelines for using EVENTACTIONS on DDL records: ■ CHECKPOINTBEFORE: Since each DDL record is autonomous, the DDL record is guaranteed to be the start of a transaction; therefore, the CHECKPOINT BEFORE event action is implied for a DDL record. ■ IGNORE: This option is not valid for DDL records. Because DDL operations are autonomous, ignoring a record is equivalent to ignoring the entire transaction. EVENTACTIONS does not support the following DDL objects because they are derived objects: ■ indexes ■ triggers ■ synonyms ■ RENAME on a table and ALTER TABLE RENAME

5.7.1 Combining DDL Parameter Options

The following is an example of how to combine DDL parameter options.

Example 5–1 Combining DDL Parameter Options

```
DDL  &
INCLUDE UNMAPPED &
   OPTYPE alter &
   OBJTYPE 'table' &
   OBJNAME users.tab* &
```

```
INCLUDE MAPPED OBJNAME * &
EXCLUDE MAPPED OBJNAME temporary.tab
```

The combined filter criteria in this statement specify the following:

- INCLUDE all ALTER TABLE statements for tables that are not mapped with a TABLE or MAP statement (UNMAPPED scope), but only if those tables are owned by users and their names start with tab,
- and INCLUDE all DDL operation types for all tables that are mapped with a TABLE or MAP statement (MAPPED scope),
- and EXCLUDE all DDL operation types for all tables that are MAPPED in scope, but only if those tables are owned by temporary and only if their names begin with tab.

5.7.2 DDL EXCLUDE ALL

DDL EXCLUDE ALL is a special processing option that maintains up-to-date object metadata for Oracle GoldenGate, while blocking the replication of the DDL operations themselves. You can use DDL EXCLUDE ALL when using a method other than Oracle GoldenGate to apply DDL to the target, but 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 may set the WILDCARDRESOLVE parameter to IMMEDIATE to allow immediate DML resolution if required.

To prevent all DDL metadata and operations from being replicated, omit the DDL parameter entirely.

5.8 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
- CREATE and DROP on an index 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*.

5.8.1 Mapping Derived Objects

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 to the target works as shown in these sections:

Section 5.8.1.1, "MAP Exists forBbase Object, But Not Derived Object"

Section 5.8.1.2, "MAP Exists for Base and Derived Objects"

Section 5.8.1.3, "MAP Exists for Derived Object, But Not Base Object"

5.8.1.1 MAP Exists forBbase Object, But Not Derived Object

If there is a MAP statement for the base object, but not for the derived object, the result is an *implicit mapping* of the derived object. Assuming the DDL parameter includes MAPPED, Replicat gives the derived object the same target owner as that of the base object. The name of the derived object stays the same as in the source statement. For example, assume the following:

Extract (source)	Replicat (target)
TABLE hr.tab*;	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:

```
CREATE INDEX hrBackup.indexPayrollDate ON TABLE hrBackup.tabPayroll (payDate);
```

The rule for the implicit mapping is based the typical industry practice of giving derived objects the same owner as the base object. Also, when indexes are owned by the same target owner as the base object, an implicit mapping eliminates the need to map derived object names explicitly.

5.8.1.2 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 parameter includes MAPPED, Replicat converts the owner and name of each object according to its own TARGET clause. For example, assume the following:

Extract (source)	Replicat (target)
TABLE hr.tab*;	MAP hr.tab*, TARGET hrBackup.*;
TABLE hr.index*;	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:

```
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 owner from that of the base object, or when the name on the target must be different from that of the source.

5.8.1.3 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.

5.8.2 New Tables as Derived Objects

The following sections explain how Oracle GoldenGate handles new tables that are created from:

- Section 5.8.2.1, "RENAME"
- Section 5.8.2.2, "CREATE TABLE AS SELECT"

5.8.2.1 RENAME

In 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.

```
RENAME hr.indexPayrollDate TO index_paydate;
```

The derived object name is hr.indexPayrollDate.

5.8.2.2 CREATE TABLE AS SELECT

CREATE TABLE AS SELECT statements include SELECT statements and INSERT statements that affect any number of underlying objects. On the target, Oracle GoldenGate obtains the data for the AS SELECT clause from the target database.

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:

```
MAP a.tab*, TARGET a.x*;
```

The target DDL statement that Replicat applies is this:

```
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.

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	Target
TABLE a.tab*;	MAPEXCLUDE a.tab2 MAP a.tab*, TARGET a.x*; MAP a.tab2, TARGET a.tab2;

5.8.3 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 owner and object names as on the source.

Table 5–3 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 5–3 [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 owner 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 the first example, shown in Table 5–4, both trigger and table are owned by rpt on the target because both base and derived names are converted by means of MAPDERIVED.

Table 5–4 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 the second example, shown in Table 5–5, the trigger is owned by fin, because conversion is prevented by means of NOMAPDERIVED.

Table 5–5 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.

5.9 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, and other things that are not directly related to data structures. 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 the following section: "How DDL is Evaluated for Processing"

5.10 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.

5.10.1 How Extract Evaluates DDL

1. Extract captures a DDL operation.
2. Extract searches for the `DDL` parameter. (This example assumes it exists.)
3. Extract gets the base object name and, if present, the derived object name.
4. 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`.

5. 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 operation and evaluates another DDL operation. 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 operation, and the processing logic continues.
6. 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 operation against each DDLSUBST statement in the parameter file. For all true DDLSUBST statements, Extract performs string substitution in the order that the DDLSUBST parameters are listed in the file.
7. Extract writes the DDL statement to the trail.

5.10.2 How Replicat Evaluates DDL

Replicat evaluates a DDL by doing the following:

1. Replicat reads the DDL operation from the trail.
2. Replicat evaluates the DDL synchronization scope to determine if the DDL qualifies for name mapping. Anything else is of OTHER scope.
3. 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.
4. Replicat replaces the source base object name with the base object name that is specified in the TARGET clause of the MAP statement.
5. 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.
6. 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 operation and starts evaluating another DDL operation. 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 operation, and the processing logic continues.
7. 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 operation against each DDLSUBST statement in the parameter file. For all true DDLSUBST statements, Replicat performs string substitution in the order that the DDLSUBST parameters are listed in the file.
8. Replicat executes the DDL operation on the target database.

9. If there are no errors, Replicat processes the next DDL statement. If there are errors, Replicat performs the following steps.
10. Replicat analyzes the INCLUDE and EXCLUDE rules in the Replicat DDLERROR parameter statements 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.
11. If the error handling does not enable the DDL operation 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.

5.11 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.

5.12 Viewing DDL Report Information

By default, Oracle GoldenGate shows basic statistics about DDL operations 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.

To view a report, use the VIEW REPORT command in GGSCI.

```
VIEW REPORT group
```

5.12.1 Extract DDL Reporting

The Extract report lists the following:

- The entire syntax of each captured DDL operation, its Oracle GoldenGate CSN number, the Teradata sequence number, 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 is an example taken from an Extract report file.

Example 5–2 Sample DDL Section from an Extract Report

```
2011-01-21 18:41:40  GGS INFO 2100  DDL found, operation [DROP TABLE "SMIJATOVDBS"."src13_tabtable_
9" ; (size 59)], start CSN [2500FF3F0200363A], DDL seqno [00000025000000000000381500000021].
2011-01-21 18:41:40  GGS INFO 2100  DDL operation included [include mapped objname "*"], optype
[DROP], objtype [TABLE], objowner [SMIJATOVDBS], objname [SRC13_TABTABLE_9].
2011-01-21 18:41:40  GGS INFO 2100  DDL operation written to extract trail file.
```

5.12.2 Replicat DDL Reporting

The Replicat report lists:

- The entire syntax of each DDL operation that Replicat processed from 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 is an example taken from a Replicat report file.

Example 5–3 Sample DDL Section from a Replicat Report

```
2011-01-21 18:41:44  GGS INFO  2104  DDL found, operation [DROP TABLE "SMIJATOVDBS"."src13_
tabtable_9" ; (size 59)].
2011-01-21 18:41:44  GGS INFO  2100  DDL is of mapped scope, after mapping new operation [DROP
TABLE "SMIJATOVDBT"."SRC13_TABTABLE_9" ; (size 59)].
2011-01-21 18:41:44  GGS INFO  2100  Executing DDL operation.
2011-01-21 18:41:44  GGS INFO  2105  DDL operation successful.
```

5.12.3 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)

5.13 Tracing DDL Processing

If you open a support case with Support, you might be asked to turn on tracing. The following parameters control DDL tracing.

- `TLTRACE` controls Extract tracing
- `TRACE` and `TRACE2` control Replicat tracing.

These parameters have options to isolate the tracing of DDL from the tracing of DML. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.

6

Modifying Objects in the Oracle GoldenGate Configuration

This chapter contains instructions for performing some common maintenance tasks when using the Oracle GoldenGate replication solution. This chapter contains the following sections:

- Section 6.1, "Deleting an Extract Group"
- Section 6.2, "Adding a Table to an Existing Extract Group"
- Section 6.3, "Moving a Table to a New Extract Group"
- Section 6.4, "Modifying Columns of a Table"

6.1 Deleting an Extract Group

To delete an Oracle GoldenGate Extract group, the Extract process must be decoupled from the Teradata replication group.

1. Start GGSCI.
2. While Extract is still running, issue this command:

   ```
   SEND EXTRACT group, VAMMESSAGE "control:terminate"
   ```

3. Stop Extract.

   ```
   STOP EXTRACT group
   ```

4. Delete the Extract group forcefully.

   ```
   DELETE EXTRACT group !
   ```

5. From any Teradata client, issue this command:

   ```
   DROP REPLICATION GROUP replication_group
   ```

6.2 Adding a Table to an Existing Extract Group

To add a table to an existing Extract group:

1. Suspend activity on the source tables that are linked to Oracle GoldenGate.
2. Start GGSCI.
3. In GGSCI, issue this command:

   ```
   INFO EXTRACT group
   ```

4. On the `Checkpoint Lag` line, verify whether there is any Extract lag. If needed, continue to issue `INFO EXTRACT` until lag is zero, which indicates that all of the transaction data so far has been processed.
5. While Extract is still running, issue this command:

   ```
   SEND EXTRACT group, VAMMESSAGE "control:terminate"
   ```

6. Stop the Extract group.

   ```
   STOP EXTRACT group
   ```

7. From any Teradata client, issue this command to add the new table:

   ```
   ALTER REPLICATION GROUP group ADD database.table
   ```

8. From any Teradata client, issue this command to generate a security token.

   ```
   ALTER REPLICATION GROUP group
   ```

9. Edit the TAM initialization file and specify the security token with the `SecurityToken` parameter.
10. Edit the Extract parameter file to add a `TABLE` parameter that specifies the new table.

    ```
    EDIT PARAMS group
    ```

11. Save and close the file.
12. In GGSCI, issue this command to start Extract:

    ```
    START EXTRACT group
    ```

13. Allow activity on the source tables that are linked to Oracle GoldenGate.

6.3 Moving a Table to a New Extract Group

To move a table to a new Extract group:

1. Suspend activity on the source database for all tables that are linked to Oracle GoldenGate.
2. Edit the current Teradata Create Group Statement file to remove the table from the `CREATE REPLICATION GROUP` statement.
3. Start GGSCI.
4. In GGSCI, issue this command for the current Extract group:

   ```
   INFO EXTRACT group
   ```

5. On the `Checkpoint Lag` line, verify whether there is any Extract lag. If needed, continue to issue `INFO EXTRACT` until lag is zero, which indicates that all of the transaction data so far has been processed.
6. In GGSCI, issue this command:

   ```
   SEND EXTRACT group, VAMMESSAGE "control:terminate"
   ```

7. Stop the current Extract group.

   ```
   STOP EXTRACT group
   ```

8. Edit the current Extract parameter file.

   ```
   EDIT PARAMS group
   ```

9. Remove the `TABLE` parameter that contains the table.
10. From any Teradata client, issue this command to drop the replication group that contains the table that is being moved:

    ```
    ALTER REPLICATION GROUP group DROP table
    ```

11. In GGSCI, issue this command to start the current Extract group, so that it can continue processing its assigned tables, minus the one that was moved:

    ```
    START EXTRACT group
    ```

12. Add a new Extract group that contains a `TABLE` statement for the moved table, and then add the other processes, trails, and parameter files that are appropriate for the capture method that you are using. See "Configuring Oracle GoldenGate" on page 4-1 for instructions.
13. Create a new `tam.ini` file and a new Teradata Create Group Statement file that contains the table. See "Configuring the TAM Initialization File" on page 3-5 and "Creating a Teradata Replication Group" on page 3-2.
14. Start the new Extract group and any associated processes.

    ```
    START EXTRACT new_group
    ```

15. Allow user activity to resume on all of the source tables that are linked to Oracle GoldenGate.

6.4 Modifying Columns of a Table

To modify columns of a table:

1. Suspend activity on the source database for all tables that are linked to Oracle GoldenGate.
2. Start GGSCI.
3. In GGSCI, issue this command for the Extract group:

   ```
   INFO EXTRACT group
   ```

4. On the `Checkpoint Lag` line, verify whether there is any Extract lag. If needed, continue to issue `INFO EXTRACT` until lag is zero, which indicates that all of the transaction data so far has been processed.
5. While Extract is still running, issue this command:

   ```
   SEND EXTRACT group, VAMMESSAGE "control:terminate"
   ```

6. Stop the Extract group.

   ```
   STOP EXTRACT group
   ```

7. In GGSCI, issue this command for the Replicat group:

   ```
   INFO REPLICAT group
   ```

8. On the `Checkpoint Lag` line, verify whether there is any Replicat lag. If needed, continue to issue `INFO REPLICAT` until lag is zero, which indicates that all of the data in the trail has been processed.

9. Stop the Replicat group.

```
STOP REPLICAT group
```

10. Perform the table modifications on the source and target databases.
11. Start the Extract and Replicat processes.

```
START EXTRACT group
START REPLICAT group
```

12. Allow user activity to resume on all of the source tables that are linked to Oracle GoldenGate.

7

Uninstalling Oracle GoldenGate

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. This chapter contains the following sections:

- Section 7.1, "Uninstalling Oracle GoldenGate from Linux or UNIX"
- Section 7.2, "Uninstalling Oracle GoldenGate from Windows (Non-cluster)"
- Section 7.3, "Uninstalling Oracle GoldenGate from Windows Cluster"

7.1 Uninstalling Oracle GoldenGate from Linux or UNIX

To uninstall Oracle GoldenGate from Linux or UNIX:

1. Run the command shell.
2. (Suggested) 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.
3. Run GGSCI.
4. While Extract is still running, issue the following command.

   ```
   SEND EXTRACT group, VAMMESSAGE "control:terminate"
   ```

5. Stop Extract.

   ```
   STOP EXTRACT group
   ```

6. Delete the Extract group forcefully.

   ```
   DELETE EXTRACT group !
   ```

7. Stop the Manager process (where ! in the following command can be used to bypass the interactive prompt).

   ```
   Stop Manager [!]
   ```

8. Exit GGSCI.
9. From any Teradata client, issue the following command.

   ```
   DROP REPLICATION GROUP repgroup name
   ```

10. Remove the Oracle GoldenGate files by removing the installation directory.

11. Drop any Oracle GoldenGate-related objects from the database as needed.

7.2 Uninstalling Oracle GoldenGate from Windows (Non-cluster)

To uninstall Oracle GoldenGate from Windows:

1. (Suggested) 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. Run GGSCI.
3. While Extract is still running, issue the following command.

   ```
   SEND EXTRACT group, VAMMESSAGE "control:terminate"
   ```

4. Stop Extract.

   ```
   STOP EXTRACT group
   ```

5. Delete the Extract group forcefully.

   ```
   DELETE EXTRACT group !
   ```

6. Stop the Manager process (where ! in the following command can be used to bypass the interactive prompt).

   ```
   STOP MANAGER [!]
   ```

7. Exit GGSCI.
8. From any Teradata client, issue the following command.

   ```
   DROP REPLICATION GROUP repgroup name
   ```

9. Click **Start** then **Run** and type `cmd` in the Run dialog box.
10. Change directories to the Oracle GoldenGate installation directory.
11. Run the INSTALL utility using the following syntax.

    ```
    install deleteevents deleteservice
    ```

    This command stops Oracle GoldenGate events from being reported to the Windows Event Manager and removes the Manager service.
12. Delete the Oracle GoldenGate installation folder.
13. Drop any Oracle GoldenGate-related objects from the database as needed.

7.3 Uninstalling Oracle GoldenGate from Windows Cluster

To uninstall Oracle GoldenGate from Windows cluster:

1. Log into the node in the cluster that owns the cluster group that contains the Manager resource. 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. Run GGSCI.
3. While Extract is still running, issue the following command.

   ```
   SEND EXTRACT group, VAMMESSAGE "control:terminate"
   ```

4. Stop Extract.

   ```
   STOP EXTRACT group
   ```

5. Delete the Extract group forcefully.

   ```
   DELETE EXTRACT group !
   ```

6. Stop the Manager process (where ! in the following command can be used to bypass the interactive prompt).

   ```
   STOP MANAGER [!]
   ```

7. Exit GGSCI.
8. Use the Cluster Administrator tool to take the Manager resource offline.
9. Right click the resource and select **Delete** to remove it.
10. Run the INSTALL utility using the following syntax.

    ```
    INSTALL DELETEEVENTS DELETESERVICE
    ```

    This command stops Oracle GoldenGate events from being reported to the Windows Event Manager and removes the Manager service.

11. Move the cluster group to the next node in the cluster, and repeat from step 10.
12. From any Teradata client, issue the following command.

    ```
    DROP REPLICATION GROUP repgroup name
    ```

13. Delete the Oracle GoldenGate installation folder.
14. Drop any Oracle GoldenGate-related objects from the database as needed.

A

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. Additional files not listed here might be installed on certain platforms. Files listed here might not be installed on every platform.

- Oracle GoldenGate Programs and Utilities
- Oracle GoldenGate Subdirectories
- Other Oracle GoldenGate Files
- Oracle GoldenGate Checkpoint Table

A.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 A–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 A–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.

A.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 A–2 Oracle GoldenGate Installed Subdirectories

Directory	Description
br	Contains the checkpoint files for the bounded recover feature.
cfg	Contains the property and XML files that are used to configure Oracle GoldenGate Monitor.
dirdb	Contains the datastore 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.

Table A–2 (Cont.) Oracle GoldenGate Installed Subdirectories

Directory	Description

`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 A–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	Used by the triggen utility to store SQL scripts before triggen was deprecated. Currently used to store 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.

A.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 A–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 recyclebin 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)
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.

Table A–3 (Cont.) Other Oracle GoldenGate Installed Files

Component	Description
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.
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)

Table A–3 (Cont.) Other Oracle GoldenGate Installed Files

Component	Description
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.

A.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*.