Oracle® GoldenGate

Teradata Installation and Setup Guide

Version 10.4

October 2009

Teradata Installation and Setup Guide, version 10.4

Contents

CHAPTER 1

System requirements and preinstallation instructions

Overview of GoldenGate for Teradata

GoldenGate supports replication of data:

- 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, GoldenGate replicates DDL operations between identical Teradata source and subscriber servers.

GoldenGate operates on a *replication server*, which is separate from the servers that contain the Teradata databases. GoldenGate receives transactional changes or table-copy operations from the Teradata Change Data Capture (CDC) facility on the source server, and then transmits it to the subscriber server using ODBC over a TCP/IP connection. Communication between the CDC and GoldenGate is managed by the Teradata Access Module (TAM).

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

What this documentation provides

This documentation contains information that is specific to the setup of the 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.

Supported Platforms

Supported database versions

- V2R6.x (DML support only)
- V12.0 (DML support only)
- V13.0 (DML and DDL support)

Supported operating systems

To find out which GoldenGate builds are available for a specific combination of database version and operating system, go to http://support.goldengate.com. A valid user name and password are required to enter this site.

Operating system requirements

Replication server

- Install GoldenGate on a server (the *replication server*) that is separate from the one where the Teradata source and target databases are installed.
  - This 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 GoldenGate Extract process on a replication server at the source location, and install the GoldenGate Replicat process on a different replication server at the target location.
- To use GoldenGate in a bidirectional Teradata configuration, you can do either of the following:
  - install GoldenGate on one replication server and use it to move data in both directions.
  - install instances of GoldenGate on separate replication servers, each one handling data movement in one direction.
- As a best practice, install GoldenGate on a multi-node cluster server to minimize the impact of a GoldenGate outage caused by server failure.
- Because of problems with the replication of Unicode data through the Linux ODBC module, use a Windows server to host GoldenGate if you will be propagating Unicode.
- Install the Teradata Access Module (TAM) library into the root GoldenGate directory on the replication server. The TAM communicates with the GoldenGate *Vendor Access Module* (VAM), the GoldenGate component that 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 GoldenGate, see the Teradata Replication Solutions documentation. In general, the TAM version should match the database version.
- For additional configuration considerations, consult the Teradata Replication Solutions documentation before installing GoldenGate.

Disk requirements

- The recommended hardware configuration for the GoldenGate server is:
  - Four 300-GB disks
  - 4 dual-core CPUs
  - 8 GB of RAM
- Assign the following free disk space:
  - 50 MB for the GoldenGate installation files. This includes space for the compressed download file and space for the uncompressed files. You can delete the download file after the installation is complete.
  - 40 MB for the working directories and binaries for each instance of GoldenGate that you are installing on the system. For example, to install two builds of GoldenGate into two separate directories, allocate 80 MB of space.
  - Additional disk space on any system that hosts GoldenGate trails, which contain the working data. The space that is consumed by the trails varies, depending on the volume of data that will be processed. A good starting point is 1 GB.
  - To install GoldenGate into a cluster environment, install the GoldenGate binaries and files on a shared file system that is available to all cluster nodes.

Relay Services Gateway (RSG) vprocs

Replication tasks run on RSG vprocs on the source 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.

TCP/IP

- Configure the system to use TCP/IP services, including DNS.
- Configure the network with the host names or IP addresses of all systems that will be hosting GoldenGate processes and to which GoldenGate will be connecting. Host names are easier to use.
- GoldenGate requires the following unreserved and unrestricted TCP/IP ports:
  - One port for communication between the Manager process and other GoldenGate processes.
  - A range of ports for local GoldenGate communications: can be the default range starting at port 7840 or a customized range of up to 256 other ports.
- Keep a record of the ports you assigned to GoldenGate. You will specify them with parameters when configuring the Manager process.
- Configure your firewalls to accept connections through the GoldenGate ports.
- If possible, grant unrestricted FTP access to GoldenGate for transfers of data, parameters, and reports between source and target systems. Otherwise, provide for another transfer method. A secure transfer method is also required to resolve support cases.
- If possible, provide a connection between your source and target systems and a site where files can be staged for transfer to and from the GoldenGate Software FTP Support Site (ftp://support.goldengate.com).

Operating system permissions

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

The Extract and Replicat processes require privileges to access the database.

Third-party programs

- Before installing GoldenGate on a Windows system, install and configure the Microsoft Visual C ++ 2005 SP1 Redistributable Package. **Make certain it is the SP1 version of this package, and make certain the get the right 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.
- GoldenGate fully supports virtual machine environments created with any virtualization software on any platform. When installing GoldenGate into a virtual machine environment, select a GoldenGate build that matches the database and the operating system of the virtual machine, not the host system.

Database requirements

Database configuration

- Install an appropriate ODBC driver:
  - 3.06.00.0x or greater for the TTU 8.2 family, V2R6.x
  - 12.00.00.01 or greater for the TTU 12.0 family, V12
  - 13.00.00.00 or greater for the TTU 13.0 family, V13
- Create Teradata replication groups for the source tables. For instructions, see the Teradata documentation.

Database user

- Create a database user that is dedicated to GoldenGate. It can be the same user for all of the GoldenGate processes that must connect to a database:
  - Extract (source database)
  - Replicat (target database)
  - DEFGEN (source or target database)
- To preserve the security of your data, and to monitor GoldenGate processing accurately, do not permit other users, applications, or processes to log on or operate as the GoldenGate database user.

- For each Teradata replication group, 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;
```

- Specify the security token with the SecurityToken=<token> parameter in the VAM.ini file. See page 40.

Supported data types

Numeric data types

- BYTEINT
- DECIMAL (NUMERIC)
- FLOAT (REAL, DOUBLE PRECISION)
- INTEGER
- SMALLINT

Limitations of support

These 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.

Single-byte character data types

- CHAR
- VARCHAR
- LONG VARCHAR

Limitations of support

These data 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.

Multi-byte character data

UNICODE and KANJISJIS multi-byte character data is supported between Teradata sources and Teradata targets. UNICODE is supported between Teradata and other databases.

Limitations of support

- Use a Windows replication server.
- Use the Teradata ODBC driver version 3.0.4 or later.
- Use the same character set on the source and target.
- Table and column names must be in ASCII.
- 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 vendor access module (VAM) to the UTF-16 character set in the initialization file (see page 40).
- When creating Replicat groups, use the NODBCHECKPOINT option with the ADD REPLICAT command. The Replicat database checkpointing feature does not work when the target ODBC driver is set to the UTF-16 character set. Checkpoints will be maintained in the checkpoint file on disk.

Binary data types

- BYTE
- VARBYTE

Limitations of support

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

Large objects

- UDT (user defined type)
- Large object data types BLOB and CLOB.

Limitations of support

- To replicate large objects, at least TAM 13.0 and GoldenGate version 10.0 are required.
- To replicate UDTs, the target database must be Teradata Database V2R6.0 or later.
- To replicate large objects from other databases to Teradata, use Teradata ODBC driver version 3.0.5 or higher on the target system. The target must support large objects delivered by ODBC.
- Enable the UseNativeLOBSupport flag in the ODBC configuration file. See the Teradata ODBC documentation.

Date data types

- DATE
- TIME
- TIMESTAMP
- TIME with TIMESZONE

- TIMESTAMP with TIMEZONE
- INTERVAL
- PERIOD

Limitations of support

- All of these data 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.

Identity data type

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.

Non-supported data types

- LARGE DECIMAL (not supported by CDC)

Supported objects and operations

DML

- GoldenGate supports the extraction and replication of DML (data manipulation language) on Teradata tables that contain rows of up to 512 KB in length.
- GoldenGate supports the maximum number of columns per table that is supported by the database.

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.

GoldenGate supports the extraction and replication of the following Teradata DDL operations, up to 2 MB in statement length. At least TAM 13.0 is required, and both source

and target databases must be Teradata Database 13.0 or later.

Table 1 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 MACRO <macro name> HASH INDEX <index name> JOIN INDEX <index name> TRIGGER <trigger name>
RENAME	TABLE <table name> TO VIEW <view name> TO MACRO <macro name> TRIGGER <trigger name>
GRANT ... ON REVOKE ... ON	TABLE <table name> 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] DROP STATISTICS ON	<table name>

[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 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.

Supported and non-supported object names and case

The following will help you verify whether the name of a supported object type qualifies or disqualifies it for inclusion in a GoldenGate configuration.

Object names and owners

Source and target object names must be fully qualified in GoldenGate parameter files, as in fin.emp.

Case sensitivity

If a database is case-sensitive, GoldenGate supports the case sensitivity of database names, owner names, object names, column names, and user names.

If a database is case-insensitive, or if it supports case-sensitivity but is configured to be case-insensitive, GoldenGate converts all names to upper case.

To preserve case-sensitivity

Case-sensitive names must be specified in GoldenGate parameter files exactly as they appear in the database. Enclose case-sensitive names in double quotes if the other database (the source or target of the case-sensitive objects) is not case-sensitive.

If replicating from a case-insensitive database to a case-sensitive database, the source object names must be entered in the Replicat MAP statements in upper case, to reflect the fact that they were written to the trail as uppercase by Extract.

For example:

```
MAP SALES.CUSTOMER, TARGET "Sales.Account";
```

Supported characters

GoldenGate supports alphanumeric characters in object names and the column names of key columns and non-key columns. GoldenGate also supports the following non-

alphanumeric characters in columns that are not being used by GoldenGate as a key.

Table 2 Supported non-alphanumeric characters in object names and non-key column names[1]

Character	Description
~	Tilde
< >	Greater-than and less-than symbols
/	Forward slash
\	Backward slash
!	Exclamation point
@	At symbol
#	Pound symbol
$	Dollar symbol
%	Percent symbol
^	Carot symbol
( )	Open and close parentheses
_	Underscore
-	Dash
+	Plus sign
=	Equal symbol
|	Pipe
[ ]	Begin and end brackets
{ }	Begin and end curly brackets (braces)

[1] The type of key that is being used by GoldenGate depends on the definition of a given table and whether there are any overrides by means of a KEYCOLS clause. GoldenGate will use a primary key, if available, or a unique key/index (selection is dependent on the database). In the absence of those definitions, all columns of the table are used, but a KEYCOLS clause overrides all existing key types. For columns that are being used by GoldenGate as a key, the characters in the names must be valid for inclusion in a WHERE clause. This list is all-inclusive; a given database platform may or may not support all listed characters.

Non-supported characters

GoldenGate does not support the following characters in object or column names:

Table 3 Non-supported characters in object and column names[1]

Character	Description
&	Ampersand
*	Asterisk
?	Question mark
:	Colon
;	Semi-colon
,	Comma
‘ ’	Single quotes
“ ”	Double quotes
‘	Accent mark (Diacritical mark)
.	Period
	Space

[1] This list is all-inclusive; a given database platform may or may not support all listed characters.

CHAPTER 2

Installing GoldenGate

Installing GoldenGate

Installation overview

These instructions are for installing GoldenGate for the first time. Installing GoldenGate installs all of the components required to run and manage GoldenGate processing (exclusive of any components required from other vendors, such as drivers or libraries) and it installs the GoldenGate utilities. The installation process takes a short amount of time.

Upgrades

To upgrade GoldenGate from one version to another, follow the instructions on the GoldenGate support site at http://support.goldengate.com.

New installations

To install GoldenGate for the first time, the following steps are required:

- Downloading GoldenGate
- Setting library paths for dynamic builds
- Installing the software

> **NOTE** Before proceeding, make certain that you have reviewed the System Requirements.

Downloading GoldenGate

1. Navigate to http://support.goldengate.com.
2. In the navigation bar, select Downloads.
3. In the navigation bar, select the platform.
4. Select the operating system and database.
5. Locate the correct GoldenGate build.
6. Click Download to transfer the software to your system.

Setting library paths for dynamic builds on UNIX

As of version 10, GoldenGate uses shared libraries. When installing GoldenGate on a UNIX system, the following must be true *before running GGSCI or any GoldenGate process*.

1. Make certain that the database libraries are added to the system's shared-library environment variables. This procedure is usually performed at database installation time. Consult your Database Administrator if you have any questions.
2. If you will be running a GoldenGate program from outside the GoldenGate installation directory on a UNIX system:
   - (Optional) Add the GoldenGate installation directory to the PATH environment variable.
   - (Required) Add the GoldenGate installation directory to the shared-libraries environment variable.

   For example, given a GoldenGate installation directory of /ggs/10.0, the second command in the following table requires these variables to be set:

Command	Requires GG libraries in environment variable?
$ ggs/10.0 > ./ggsci	No
$ ggs > ./10.0/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 following:

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

[1] A specific platform may or may not be supported by GoldenGate for your database. See the Systems Requirements for supported platforms.

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

NOTE To view the libraries that are required by a GoldenGate process, use the ldd <process> shell command before starting the process. This command also shows an error message for any that are missing.

Installing GoldenGate on Linux and UNIX

Installing into a UNIX or Linux cluster

- To install GoldenGate into a cluster environment, install the GoldenGate binaries and files on a file system that is available to all cluster nodes, according to the directions that follow.
- After installing GoldenGate, configure the GoldenGate Manager process within the cluster application, as directed by the cluster documentation, so that GoldenGate will fail over properly with the other applications.

Installing the GoldenGate files

1. FTP the file in binary mode to the system and directory where you want GoldenGate to be installed.
2. Extract the gzipped tar file (use the gzip or tar options appropriate for your system). The files are placed in the current directory. If gzip is not installed, unzip the file on a Windows system by using WinZip or an equivalent compression product, and then FTP the file in binary format to the installation machine.

   ```
   gzip -dc <filename>.tar.gz | tar -xvof -
   ```

   This is an example:

   ```
   gzip -dc sun29_ora102_v9527_007.tar.gz | tar -xvof -
   ```

3. Run the command shell and change directories to the new GoldenGate directory.

4. From the GoldenGate directory, run the GGSCI program.

   ```
   GGSCI
   ```

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

   ```
   CREATE SUBDIRS
   ```

6. Issue the following command to exit GGSCI.

   ```
   EXIT
   ```

Configuring Manager and other processes

- To use GoldenGate, you must configure the Manager process. You must specify a TCP/IP port for Manager to use, and you can specify optional parameters that control dynamic port assignments, trail file maintenance, and other properties.
- To begin using GoldenGate, you need to create and configure at least one Extract and Replicat group. Your instructions for these groups determine which data to capture and replicate, and how that data is processed.
- To configure these processes, and to customize GoldenGate, see the *GoldenGate for Windows and UNIX Administrator Guide*.

Installing GoldenGate on Windows and Windows Cluster

Installing GoldenGate into a Windows Cluster

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

Installing the GoldenGate files

1. Unzip the downloaded file(s) using PKUNZIP or WinZip.
2. Move the files in binary mode to a folder on the drive where you want to install GoldenGate. *Do not* install GoldenGate into a folder that contains spaces in its name, for example "GoldenGate Software." GoldenGate relies on path names, but the operating system does not support path names that contain spaces, whether or not they are within quotes.
3. From the GoldenGate folder, run the GGSCI program.

   ```
   GGSCI
   ```

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

   ```
   CREATE SUBDIRS
   ```

5. Issue the following command to exit GGSCI.

   ```
   EXIT
   ```

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, such as one for the GoldenGate replication software and one for GoldenGate Veridata. Each Manager on a system must have a unique name. Before proceeding further, verify 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
   ```

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 referenced during installation of the Windows service and during data processing.

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** > **Run**, and type **cmd** in the **Run** dialog box.
3. From the directory that contains the Manager program that you are installing as a service, run the **install** program with the following syntax:

   ```
   install <option> [...]
   ```

Where: <option> is one of the following:

Table 4 INSTALL options

Option	Description
ADDEVENTS	Adds GoldenGate events to the Windows Event Manager. By default, GoldenGate errors are generic. To produce more specific error content, copy the following files from the GoldenGate installation directory to the SYSTEM32 directory. category.dll ggsmsg.dll
ADDSERVICE	Adds Manager as a service by the name specified 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	Specifies that the service created with ADDSERVICE is to be started at system boot time. This is the default unless MANUALSTART is used.
MANUALSTART	Specifies that the service created with ADDSERVICE is to be started manually through GGSCI, a script, or the Services applet of the Control Panel. The default is AUTOSTART.
USER <name>	Specifies a domain user account for executing Manager. For <name>, include the domain name, a backward slash, and the user name, for example HEADQT\GGSMGR. By default, the Manager service is installed to use the Local System account.
PASSWORD <password>	Specifies the password for the user specified with USER.

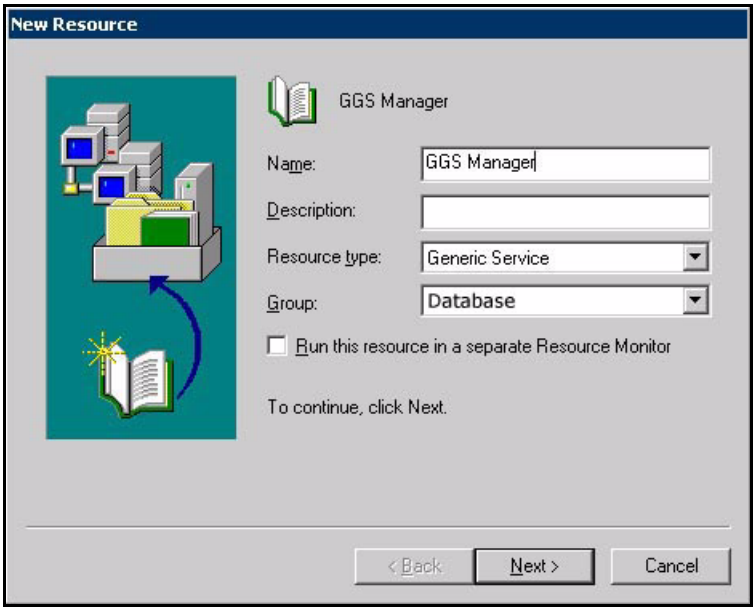
[1] A user account can be changed by selecting the Properties action from the Services applet of the Windows Control Panel.

Adding GoldenGate as a Windows cluster resource

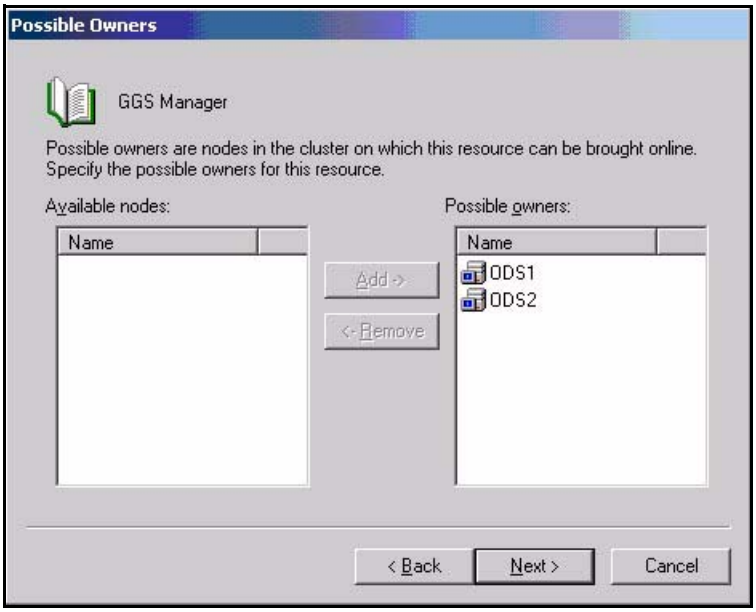
If you installed GoldenGate into a cluster, follow these instructions to establish GoldenGate as a cluster resource and configure the Manager service correctly on all nodes.

1. In the Cluster Administrator, select **File>New>Resource**.

2. In the New Resource dialog box, provide a descriptive name for the GoldenGate Manager (need not be its actual name). For Resource Type, select Generic Service. For Group, select the group that contains the database instance to which GoldenGate will connect.

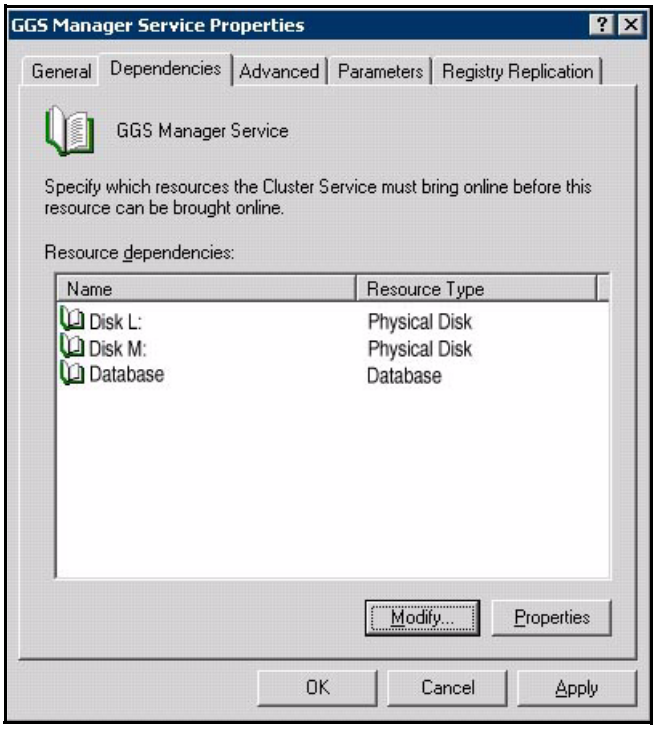


3. Click **Next**.
4. In the Possible Owners dialog box, select the nodes on which GoldenGate will run.

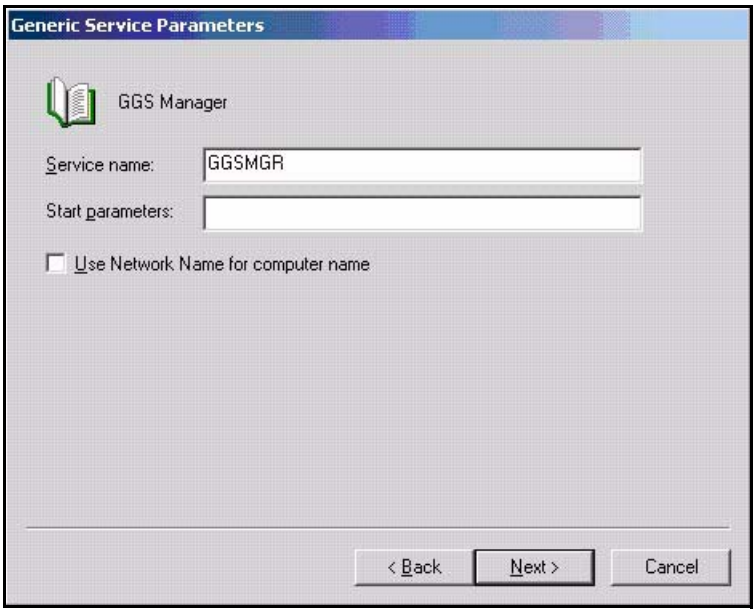


5. Click **Next**.
6. In the GGS Manager Service Properties dialog box, click the Dependencies tab, and add the following to the Resource dependencies list:
   - The database resource group (in this example, it is "Database")

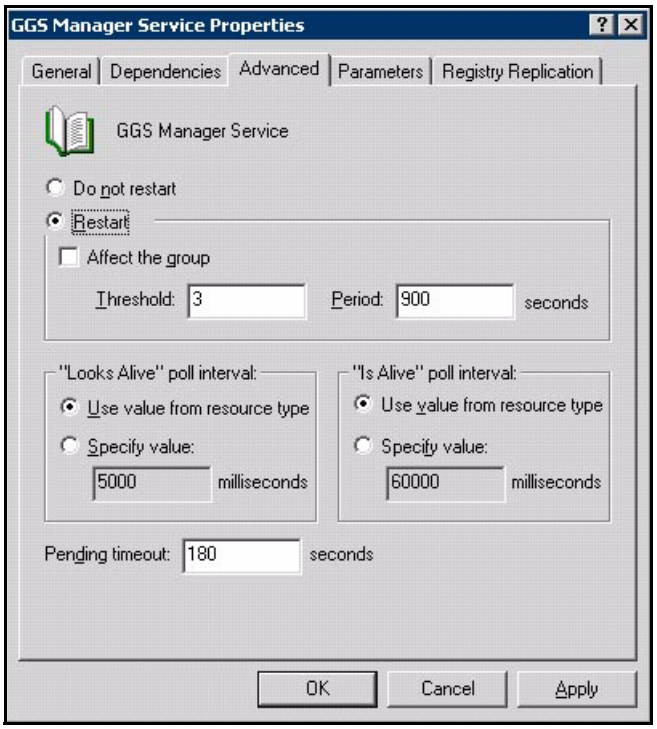
- The disk resource containing the GoldenGate directory
- The disk resource containing the database transaction log files
- The disk resource containing the database transaction log backup files



7. Click **Apply**, then **OK**.
8. In the Generic Service Parameters dialog box, type either the default Manager service name of GGSMGR or, if applicable, the custom name specified in the GLOBALS file.



9. Click **Next**.
10. Click **Finish** to exit the wizard.
11. In the Cluster Administrator tree, right-click the Manager resource and select Properties.
12. Click the Advanced tab, and deselect Affect the Group. This is a recommendation, but you can configure it as needed for your environment.



13. Click **Apply**.
14. Bring the cluster resource online to verify that it was installed correctly.
15. Take the resource offline again.
16. Move the group to the next node in the cluster. When the group has been successfully moved to the second node, the Manager resource should still be offline.
17. Log onto the second node.
18. Install GoldenGate Manager as a service on this node by running the **install** program as you did on the previous node. If you created a custom name for Manager in the GLOBALS file, that name will be used.
19. Bring the resource online to verify that it is running correctly on this node.
20. Repeat steps 18 through 22 for each additional node in the cluster.

Configuring Manager and other processes

- To use GoldenGate, you must configure the Manager process. You must specify a TCP/IP port for Manager to use, and you can specify optional parameters that control dynamic port assignments, trail file maintenance, and other properties.
- To begin using GoldenGate, you need to create and configure at least one Extract and Replicat group. Your instructions for these groups determine which data to capture and replicate, and how that data is processed.
- To configure these processes, and to customize GoldenGate, see the *GoldenGate for Windows and UNIX Administrator Guide*.

CHAPTER 3

Preparing the system for GoldenGate

Preparing tables for processing

The following table attributes must be addressed in a GoldenGate environment.

Assigning row identifiers

GoldenGate requires some form of key on the source and target tables to identify and locate the correct target rows for replicated updates and deletes.

Determining a key to use

A key can be one of the following:

- Primary key or unique primary index

  If possible, create or alter source and target tables to have a primary key or a unique primary index. With one of these identifiers on the source and a matching one on the target, GoldenGate can locate the required target row with an efficient WHERE clause.

- Substitute key

  If a primary key or unique primary index cannot be added to source and target tables, find out whether the table has columns that always contain unique values. You can specify those columns in a KEYCOLS clause within the Extract TABLE parameter and the Replicat MAP parameter. GoldenGate will use those columns as a substitute key.

- All-column key

  In the absence of a primary key, a unique primary index, or KEYCOLS columns, GoldenGate constructs a pseudo key by using all of the columns of the table, excluding certain data types and columns excluded from the GoldenGate configuration. Constructing this key impedes the performance of GoldenGate on the source system. On the target, this key causes Replicat to use a larger, less efficient WHERE clause.

Disabling triggers and cascade delete constraints

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

Configuring the ODBC driver

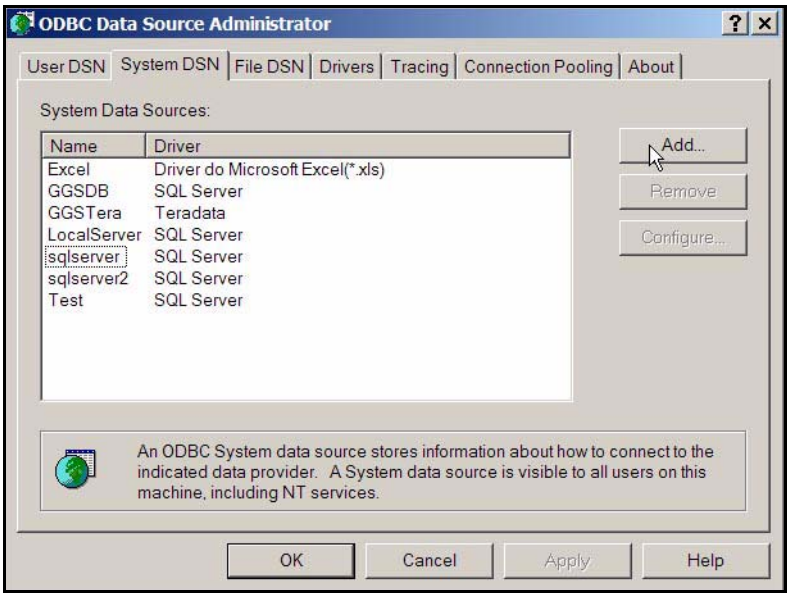
Follow these direction to configure ODBC (Open Database Connectivity).

Creating a Teradata DSN

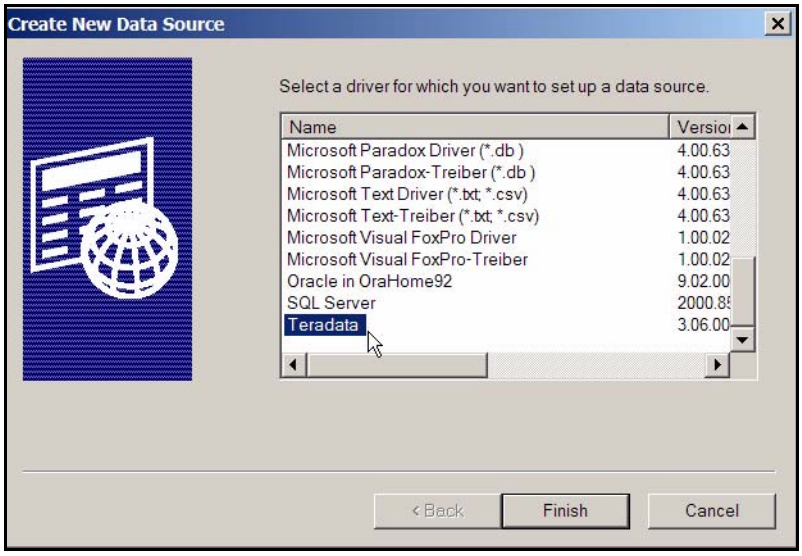
Establish a system data source name (DSN) on each source and target system where GoldenGate will interface with a Teradata database. A DSN stores information about how to connect to the database.

To create a Teradata DSN on Windows

1. Click **Start > Settings > Control Panel**.
2. Double-click **Administrative Tools**.
3. Double-click **Data Sources (ODBC)** to open the **ODBC Data Source Administrator** dialog box.
4. Select the **System DSN** tab, and then click **Add**.

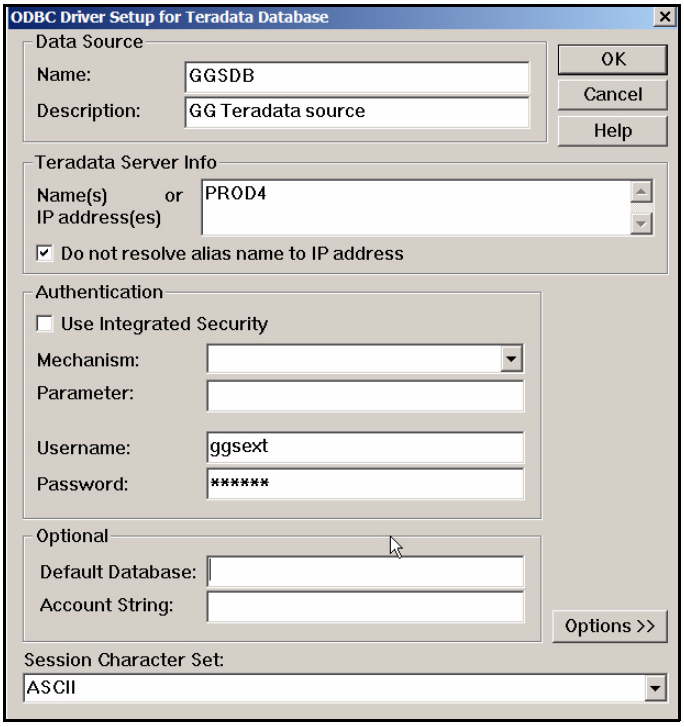


5. Under **Create New Data Source**, select the Teradata driver.

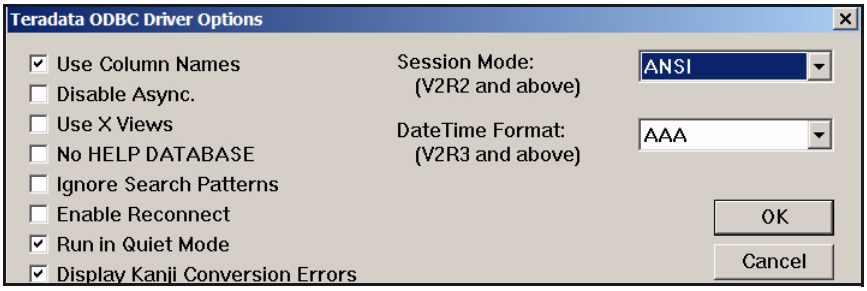


6. Click **Finish**. The **Create a New Data Source to Teradata** wizard is displayed.
7. Supply the following:
   - **Name**: Can be of your choosing. In a Windows cluster, use one name across all nodes in the cluster.
   - **Description**: (Optional) Type a description of this data source.
   - **Teradata Server Info**: Supply the name of the Teradata server.
   - **Authentication**: Supply the database authentication that the GoldenGate process will be using.
   - **Optional**: Set the default database to the one that GoldenGate will be connecting to.

- **Session Character Set**: Select ASCII.

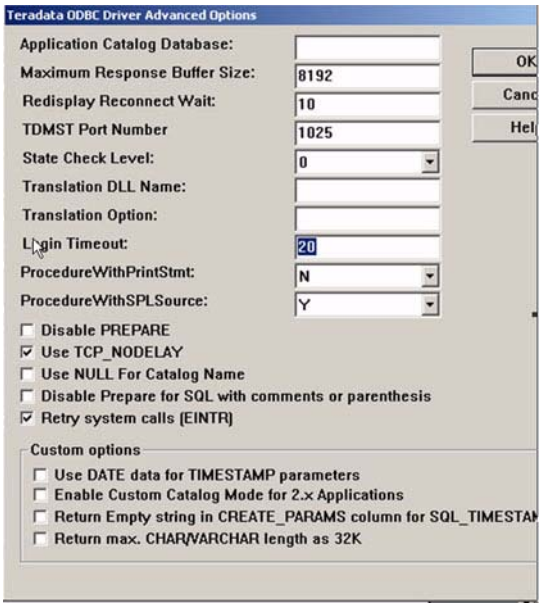


8. On the same page, click **Options** to view the **Teradata ODBC Driver Options**.
   - **Session Mode**: Select ANSI.
   - **DateTime Format**: Select AAA.
   - Leave the other options set to their defaults.



9. Click **Advanced** to view the **Teradata ODBC Driver Advanced Options**.

10. Set the Login Timeout parameter to be greater than the default of 20.



11. Click **OK** to close each dialog until you exit the wizard.
12. Repeat these steps on the other systems where GoldenGate will interact with Teradata.

To create a Teradata DSN on UNIX or Linux

1. Use a text editor to create a text file.
2. Enter the following information. Refer to the sample .odbc.ini file in Figure 1.
   - In the [ODBC] section, specify the installation directory of the Teradata ODBC driver. The typical location is /usr/odbc. For more information, see the *Teradata ODBC for Unix* guide.
   - In the [ODBC Data Sources] section, list data source names that will be defined in the file.
   - In the remainder of the file, define the data sources that you listed under [ODBC Data Sources]. Use [default] to list default data source settings for data sources not listed under ODBC Data Sources (not shown in the example).
   - Set the LoginTimeout parameter to a value that is greater than 20.

Figure 1 Sample .odbc.ini file

```
# Teradata ODBC data source specifications
# Teradata ODBC install directory (required).
# Optionally specify trace settings.
[ODBC]
InstallDir=/usr/odbc
# List of data sources and drivers defined in this file. If a requested
# data source is not listed, the [default] will be used.
[ODBC Data Sources]
<dsn>=tdata.so
# The ODBC driver settings. Driver path and DBCName are required;
# other settings optional.
[<dsn>]
Driver=/usr/odbc/drivers/tdata.so
Description=Generic ODBC to <server> v6
DBCName=<server>
# Username/password to connect. If password expires, update this file.
Username=ggstera
Password=ggs1678
# Default database to use, if none specified.
DefaultDatabase=<database>
Database=<database>
# For GoldenGate, it is recommended to set the SessionMode and
# Time format to ANSI, unless directed otherwise.
SessionMode=ANSI
# Set DATE, TIME, and TIMESTAMP (respectively) to ANSI.
DateTimeFormat=AAA
# Driver should not display error messages on screen
RunInQuietMode=Yes
# Driver should not parse SQL; rather, send directly to Teradata
NoScan=Yes
# Login timeout should be higher than 20 seconds to avoid timeouts.
LoginTimeout=40
```

3. Save the file as .odbc.ini in the home directory of the user that executes the GoldenGate processes (for example /home/gguser/.odbc.ini). To store this file in a different location, set the ODBCINI environment variable to the full path name of the file, for example:

```
ODBCINI=/dir1/dir2/.odbc.ini; export ODBCINI
```

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.

Improving Replicat performance over ODBC

To improve the throughput of the Replicat process, consider using multiple trails associated with parallel Replicat processes. Because each Replicat uses a single ODBC session, multiple sessions might be necessary to achieve reasonably high delivery rates.

Choosing and configuring an Extract commit mode

GoldenGate works with the Change Data Capture (CDC) component of a source Teradata database to operate in one of two modes:

- *maximum protection mode*
- *maximum performance mode*

The mode that is used determines the commit protocol that is used and whether or not GoldenGate has an effect on the Teradata applications.

Maximum protection mode

Maximum protection mode is the recommended GoldenGate configuration for the Teradata database. Maximum protection mode incorporates 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 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 that it saved the data to a GoldenGate VAM trail on disk.

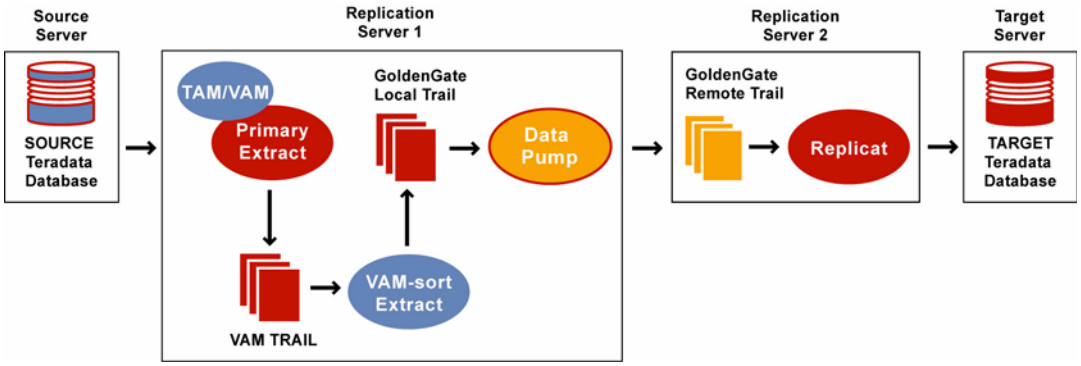
- If CDC receives the acknowledgement within a given timeout period, it releases the transaction for commit to the application, for commit to the database, and for propagation by GoldenGate.
- If CDC does not receive the acknowledgement within a given timeout period, it rolls back the transaction, and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

Recommended maximum protection configuration

It is the best practice to install the Extract and Replicat processes on separate replication servers and to use a data pump with a local trail on the server where the Extract processes are installed. In this configuration, the VAM-sort Extract persists the sorted data to a regular 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 2 Recommended maximum protection configuration



To configure Extract in maximum protection mode

Perform these steps on the source replication server.

1. Configure the Manager process according to the instructions in the *GoldenGate for Windows and UNIX Administrator Guide*.
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 following parameters plus any others that apply to your database environment.

   ```
   -- 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
   -- Specify VAM library, initialization file, and other parameters:
   VAM <library>, PARAMS (“<init file>”, “...>”)
   -- 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 following parameters plus any others that apply to your database environment.

   ```
   -- 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 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 following parameters plus any others that apply to your database environment.

```
-- 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.

To configure Replicat

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

1. Configure the Manager process according to the instructions in the *GoldenGate for Windows and UNIX Administrator Guide*.
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. This is a best practice. There are multiple options for this purpose. For instructions, see the *GoldenGate for Windows and UNIX Administrator Guide*.
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 following parameters plus any others that apply to your database environment.

```
-- 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>];
```

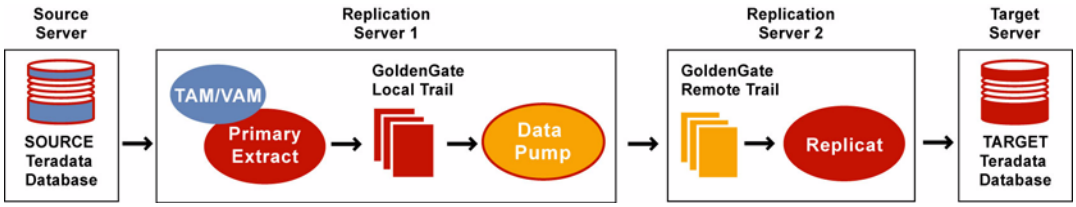
Maximum performance mode

Maximum performance mode is faster and less intrusive than maximum protection mode, but 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.

It is best practice to install the Extract and Replicat processes on separate replication servers, and to use a data pump with a local trail on the Extract server. In this configuration, the primary Extract persists the transaction to a local 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 longlasting outage.

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.

Figure 3 Recommended maximum performance configuration



To configure Extract in maximum performance mode

Perform these steps on the source replication server.

1. Configure the Manager process according to the instructions in the *GoldenGate for Windows and UNIX Administrator Guide*.

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 following parameters plus any others that apply to your database environment.

   ```
   -- Identify the Extract group:
   EXTRACT <ext>
   -- 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
   -- Specify VAM library, initialization file, and other parameters:
   VAM <library>, PARAMS (“<init file>”, “...>”)
   -- 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 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 following parameters plus any others that apply to your database environment.

```
-- 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.

To configure Replicat

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

1. Configure the Manager process according to the instructions in the *GoldenGate for Windows and UNIX Administrator Guide*.
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. This is a best practice. There are multiple options for this purpose. For instructions, see the *GoldenGate for Windows and UNIX Administrator Guide*.
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 following parameters plus any others that apply to your database environment.

```
-- 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>];
```

Creating a Teradata replication group

To create a replication group, it is best practice to use a Create Group Statement file. By using a Create Group Statement file, the correct identifier information for the replication group is automatically written to the tam.ini file. For more information, see the Teradata 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 GoldenGate Extract group.
3. Save the file with the suffix .sql in a directory named dirtam within the GoldenGate directory.
4. Specify the name of this file with the CreateGroupStmtFile parameter in the TAM initialization file. See "Configuring the initialization file" for more information.

Figure 4 Sample Create Group Statement file

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

Activating DDL capture by the Teradata TAM

To specify DDL that you want the Teradata TAM to pass to 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 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 Solutions documentation.

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 <rule set name>
[, DEFAULT]
FOR <replication group name>
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>] ]
```

Where: <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 documentation.

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 disable replication of DDL

Log in as a user with REPLCONTROL privilege and issue either of the following commands:

To disable the DDL but keep the replication group:

```
DROP REPLICATION RULESET <rule set name> FOR <replication group name>
```

To disable the DDL and delete the replication group:

```
DROP REPLICATION GROUP <group name> [ ( <table name> [, ... ] )]
```

Example `DROP REPLICATION RULESET Sales1 FOR MyRepGroup;`

Example `DROP REPLICATION GROUP MyRepGroup`

To configure GoldenGate DDL replication, see the *GoldenGate for Windows and UNIX Administrator Guide*.

Configuring the 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.

To create an initialization file

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. See Figure 5 on page 44 for an example.

Table 5 Required initialization file parameters

Parameter	Description
`Mode={Replication | TableCopy}`	**Required** The type of GoldenGate extraction to be performed. ◆ Replication specifies ongoing change data capture. ◆ TableCopy specifies full table capture for an initial data load. To configure Extract for either of these modes, see the *GoldenGate for Windows and UNIX Administrator Guide*.

Table 5 Required initialization file parameters (continued)

Parameter	Description
`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.
`ReplicationGroupName=<name>`	**Required for versions earlier than TAM12** The name of the replication group, as specified in the CREATE REPLICATION GROUP statement.
`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 <rep group name>;`
	For example:
	`help replication group g1;`
	The ID is a numerical ID in the Identifier column next to the name of the group.

`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.

Table 5 Required initialization file parameters (continued)

Parameter	Description
`NumTableCopyStreams=<n>`	**Required for TableCopy mode** The number of data streams. Ideally, there should be as many data streams as RSGs. Ignored for Replication mode.
`MaxCopyQueueSize=<n>`	**Required for TableCopy mode** The maximum number of data blocks that the data threads can store in the copy queue. This value should be small (around 4) when there are numerous streams (20 or more) and larger (about 12) when there are few streams (4 or fewer). Ignored for Replication mode.
`CopyThreadDataWaitSleep=` `<seconds>`	**Required for TableCopy mode** The number of seconds, expressed as a whole number or with decimals, that a copy thread sleeps after a read attempt where no data was present. Use a smaller number, for example .1, with four or fewer threads, or use a higher number, for example .5, for five or more threads. Ignored for Replication mode.
`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.

`CopyCheckpointInterval=` `<# rows>`	**Required for TableCopy mode** The number of rows to process between checkpoints during a table copy operation. Because the copy succeeds or fails as a whole, there is no advantage to frequent checkpointing, so a large number is appropriate. Ignored for Replication mode.

Table 5 Required initialization file parameters (continued)

Parameter	Description
`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=<0-60 secs>`	**Optional** The timeout, in seconds, when polling RSG for data. The default is 1.
`RsgTimeoutMSec= <0-60000000 millisecs>`	**Optional** The timeout, in milliseconds, when polling RSG for data. The default is 0.
`Tracing= { Debug | Performance | All | None}`	**Optional** The level of debug tracing. The default is None.
`MaxProtTransCompleteThresh= <0-24 transactions>`	**Optional, valid for max protection mode** The number of outstanding transactions that can be held after which a GoldenGate checkpoint must be requested. This is significant when operating in maximum protection mode. If there are a large number of sessions 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.
`Bidirectional {TRUE | FALSE}`	**Optional** Specifies whether or not before images of data are sent to 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 in the dirsql sub-directory within the GoldenGate directory.
4. Specify the name of this file with the VAM parameter in the Extract parameter file.

Figure 5 Sample 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
# GroupID=1008
# SecurityToken=EF4B77783C9EEA57
# NumTableCopyStreams=7
# MaxCopyQueueSize=7
# CopyThreadDataWaitSleep=0.5
CharacterSet=ASCII
ControlRSG=10.10.10.49:1152
DataRSG1=10.10.10.50:1153
DataRSG2=node3
DataRSG3=node4:1155
CopyCheckpointInterval=32767
Encryption=All
```

Handling errors on multiset tables

Handling "record not found" errors

If a replication group contains multiset tables, you might need to configure error handling for them if you expect to have duplicate rows. Otherwise, when the duplicate rows are updated or deleted, Replicat will process one operation for each of the duplicate rows. On the subscriber, the first of those operations will update or delete all of the duplicates, causing the subsequent operation to return a "record not found" error. To prevent Replicat from abending on these errors, you can include the following `REPERROR` parameters in the Replicat parameter file to ignore and log the operation:

```
REPERROR DEFAULT, ABEND
REPERROR 100, DISCARD
```

NOTE You can use this parameter at the `MAP` level for a specific table, or you can use it globally at the root level of the parameter file. If used globally, it will apply the error-handling rules in the same way for all "record not found" errors, which could be errors caused by something other than multiset tables.

Handling duplicate rows caused by Multiload

Multiload does not participate in the full two-phase commit protocol of maximum protection mode. Thus, 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 to ignore duplicate rows.

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:

- Use "set session override replication on;commit;" to perform the operations without replication on the source and target systems.
- Set the Replicat parameter MAXTRANSOPS to a value of less than 1000. This parameter splits large transactions into smaller ones.

Performing initial synchronization

Perform an initial synchronization of the source and target data before using 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.
- Small, static tables, such as postal-code lookup tables, can be copied to the target database by using the Teradata table copy feature through GoldenGate.

To configure an initial load, see the *GoldenGate for Windows and UNIX Administrator Guide*.

CHAPTER 4

Modifying objects in the GoldenGate configuration

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

Deleting an Extract group

To delete a 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 <repgroup name>
   ```

Adding a table to an existing Extract group

1. While Extract is still running, edit the Extract parameter file.

   ```
   EDIT PARAMS <group>
   ```

2. Add a `TABLE` parameter to specify the new table.
3. Save and close the parameter file.
4. Suspend activity on the source database for all tables that are linked to GoldenGate.
5. Start GGSCI.
6. In GGSCI, issue this command:

   ```
   INFO EXTRACT <group>
   ```

7. On the Checkpoint Lag line, verify that Extract has no lag.
8. Stop the Extract group.

   ```
   STOP EXTRACT <group>
   ```

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

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

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

    ```
    ALTER REPLICATION GROUP <group>
    ```

11. Edit the initialization file and specify the security token with the SecurityToken parameter.
12. In GGSCI, issue this command to start Extract:

    ```
    START EXTRACT <group>
    ```

13. Allow activity on the source database for all tables that are linked to GoldenGate.

Moving a table to a new Extract group

1. While Extract is still running, edit the Extract parameter file.

   ```
   EDIT PARAMS <group>
   ```

2. Remove the TABLE parameter that contains the table.
3. Create a new Extract parameter file.
4. Edit the Teradata Create Group Statement file to remove the table from the CREATE REPLICATION GROUP statement.
5. Create a new tam.ini file and a new Teradata Create Group Statement file.
6. Suspend activity on the source database for all tables that are linked to GoldenGate.
7. Start GGSCI.
8. In GGSCI, issue this command:

   ```
   INFO EXTRACT <group>
   ```

9. On the Checkpoint Lag line, verify that Extract has no lag.
10. In GGSCI, issue this command:

    ```
    SEND EXTRACT <group>, vammessage "control:terminate"
    ```

11. Stop the Extract group.

    ```
    STOP EXTRACT <group>
    ```

12. From any Teradata client, issue this command:

    ```
    drop replication group <repgroup name>
    ```

13. In GGSCI, issue this command to start Extract:

```
START EXTRACT <group>
```

14. Add the new Extract group with the ADD EXTRACT command. See the *GoldenGate for Windows and UNIX Reference Guide* if you are unsure which command options to use.

```
ADD EXTRACT <group> <options>
```

15. Start the new Extract group.

```
START EXTRACT <group>
```

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

Modifying columns of a table

1. Start GGSCI.
2. In GGSCI, issue this command for the Extract group:

```
INFO EXTRACT <group>
```

3. On the Checkpoint Lag line, verify that Extract has no lag.
4. Stop the Extract group.

```
STOP EXTRACT <group>
```

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

```
INFO REPLICAT <group>
```

6. On the Checkpoint Lag line, verify that Replicat has no lag.
7. Stop the Replicat group.

```
STOP REPLICAT <group>
```

8. Perform the table modifications on the source and target databases.
9. Start the Extract and Replicat processes.

```
START EXTRACT <group>
START REPLICAT <group>
```

CHAPTER 5

Uninstalling GoldenGate

This procedure assumes that you no longer need the data in the GoldenGate trails, and that you no longer need to preserve the current GoldenGate environment. To preserve your current environment and data, make a backup of the GoldenGate directory and all subdirectories before starting this procedure.

Uninstalling 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 GoldenGate commands, and to delete files and directories from the operating system.
3. Change directories to the GoldenGate installation directory.
4. Run GGSCI.
5. Stop all GoldenGate processes.
6. Stop the Manager process.
7. Exit GGSCI.
8. Remove the GoldenGate files by removing the installation directory.
9. Drop any GoldenGate-related objects from the database as needed.

Uninstalling GoldenGate from Windows (non-cluster)

1. (Suggested) Log on as the system administrator, or as a user with permission to issue GoldenGate commands, and to delete files and directories from the operating system.
2. From the GoldenGate installation folder, run GGSCI.
3. Stop all GoldenGate processes.
4. Stop the Manager program or service.
5. Exit GGSCI.
6. Click **Start** > **Run**, and type cmd in the **Run** dialog box.
7. Change directories to the GoldenGate installation directory.

8. Run the install program using the following syntax.

   ```
   install deleteevents deleteservice
   ```

   This command deletes GoldenGate events from being reported to the Windows Event Manager and removes the GoldenGate Manager service.
9. Delete the CATEGORY.DLL and GGSMSG.DLL files from the Windows SYSTEM32 folder.
10. Delete the GoldenGate installation folder.
11. Drop any GoldenGate-related objects from the database as needed.

Uninstalling GoldenGate from Windows Cluster

1. Working from the node in the cluster that owns the cluster group containing the Manager resource, run GGSCI and then stop any Extract and Replicat processes that are still running.
2. Use the Cluster Administrator tool to take the Manager resource offline.
3. Right click the resource and select **Delete** to remove it.
4. Run the install program using the following syntax.

   ```
   install deleteevents deleteservice
   ```

   This command deletes GoldenGate events from being reported to the Windows Event Manager and removes the GoldenGate Manager service.
5. Delete the CATEGORY.DLL and GGSMSG.DLL files from the Windows SYSTEM32 folder.
6. Move the cluster group to the next node in the cluster, and repeat from step 4.
7. Delete the GoldenGate installation folder.
8. Drop any GoldenGate-related objects from the database as needed.

APPENDIX 1

GoldenGate Components

This appendix describes the programs, directories, and other components created or used by the GoldenGate software in the GoldenGate installation directory. Additional files not listed here might be installed on certain platforms. Files listed here might not be installed on every platform.

GoldenGate Programs and Utilities

This section describes programs installed in the root GoldenGate installation directory.

Table 6 Programs and utilities

Program	Description
cobgen	Generates source definitions based on COBOL layouts. Used for GoldenGate for Datawise on Stratus.
convchk	Converts checkpoint files to a newer version.
ddlcob	Generates target DDL table creation statements based on COBOL layouts. Used for GoldenGate for Datawise on Stratus.
ddlgen	Generates target database table definitions based on source database DDL.
defgen	Generates data definitions and is referenced by 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 extraction from database tables or transaction logs or receives transaction data from a vendor access module.
ggmxinstall	GoldenGate installation script for SQL/MX.
ggsci	User interface to GoldenGate for issuing commands and managing parameter files.

Table 6 Programs and utilities (continued)

Program	Description
ggsmgr.jcl ggsmgr.proc ggsmgrst.jcl ggsmgrst.proc	Start the GoldenGate Manager process from a batch job or the operator console on a z/OS system.
install	Installs 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 GoldenGate processes, reporting, and routing of requests through the GGSCI interface.
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.
triggen	Generates scripts that create the GoldenGate log table and logging triggers to support the trigger-based extraction method.
vamserv	Started by Extract to read the TMF audit trails generated by TMF-enabled applications using the NonStop SQL/MX database.

GoldenGate subdirectories

This section describes the subdirectories of the GoldenGate installation directory and their contents.

Table 7 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 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
dirdat	The default location for GoldenGate trail files and extract files created by Extract processes to store records of extracted data for further processing, either by the Replicat process or another application or utility. Written in internal GoldenGate format. 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 GoldenGate technical support. Example: defs.dat
dirout	This directory is not used any more.

Table 7 Subdirectories (continued)

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 GoldenGate parameter files created by GoldenGate users to store run-time parameters for GoldenGate process groups or utilities. Written in external ASCII format. File name format is <group name/user-defined name>.prm or mgr.prm. These files may be edited to change GoldenGate parameter values. They can be edited directly from a text editor or by using the EDIT PARAMS command in GGSCI. Examples: defgen.prm finance.prm
dirrec	Not used by 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	The default location for scripts created by the TRIGGEN utility to contain SQL syntax for creating GoldenGate logging triggers and GoldenGate log tables. Written in external ASCII format. File name format is a user-defined name or the defaults of GGSLOG (table-creation script) or the table name (trigger-creation script), with the extension of .sql. These scripts can be edited if needed. Examples: ggslog.sql account.sql

Table 7 Subdirectories (continued)

Directory	Description
dirtmp	The default location for storing large transactions when the size exceeds the allocated memory size. Do not edit these files.
dirver	A GoldenGate Veridata directory. Not used unless this software is installed in the GoldenGate location.

Other GoldenGate files

This section describes other files, templates, and other objects created or installed in the root GoldenGate installation directory.

Table 8 Other files

Component	Description
bcpfmt.tpl	Template for use with Replicat when creating a run file for the Microsoft BCP/DTS bulk-load utility.
blowfish.txt	Blowfish encryption software license agreement.
category.dll	Windows dynamic link library used by the INSTALL program.
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_access.tpl	Template used by the DDLGEN utility to convert source DDL to Microsoft Access DDL.
ddl_cleartrace.sql	Script that removes the DDL trace file. (Oracle installations)
ddl_db2.tpl	Template used by the DDLGEN utility to convert source DDL to DB2 DDL (Linux, UNIX, Windows).
ddl_db2_os390.tpl	Template used by the DDLGEN utility to convert source DDL to DB2 DDL (z/OS systems).
ddl_disable.sql	Script that disables the GoldenGate DDL trigger. (Oracle installations)
ddl_enable.sql	Script that enables the GoldenGate DDL trigger. (Oracle installations)

Table 8 Other files (continued)

Component	Description
ddl_informix.tpl	Template used by the DDLGEN utility to convert source DDL to Informix DDL.
ddl_mss.tpl	Template used by the DDLGEN utility to convert source DDL to SQL Server DDL.
ddl_mysql.tpl	Template used by the DDLGEN utility to convert source DDL to MySQL DDL.
ddl_nssql.tpl	Template used by the DDLGEN utility to convert source DDL to NonStop SQL DDL.
ddl_ora9.sql	A script that gets tablespace information from an Oracle 9 database.
ddl_ora10.sql	A script that disables the Oracle recyclebin and gets tablespace information from an Oracle 10 database.
ddl_oracle.tpl	Template used by the DDLGEN utility to convert source DDL to Oracle DDL.
ddl_pin.sql	Script that pins DDL tracing, the DDL package, and the DDL trigger for performance improvements. (Oracle installations)
ddl_remove.sql	Script that removes the DDL extraction trigger and package. (Oracle installations)
ddl_setup.sql	Script that installs the GoldenGate DDL extraction and replication objects. (Oracle installations)
ddl_sqlmx.tpl	Template used by the DDLGEN utility to convert Tandem Enscribe DDL to NonStop SQL/MX DDL.
ddl_status.sql	Script that verifies whether or not each object created by the GoldenGate DDL support feature exists and is functioning properly. (Oracle installations)
ddl_sybase.tpl	Template used by the DDLGEN utility to convert source DDL to Sybase DDL.
ddl_tandem.tpl	Template used by the DDLGEN utility to convert source DDL to NonStop SQL DDL.
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.

Table 8 Other files (continued)

Component	Description
demo_<db>_create.sql	Script that creates demonstration tables in the database associated with the GoldenGate installation.
demo_<db>_insert.sql	Script that inserts initial test data into the demonstration tables.
demo_<db>_misc.sql	Script that simulates transaction activity on the demonstration tables.
ENCKEYS	User-created file that stores encryption keys. Written in external ASCII format.
exitdemo.c	User exit example.
ggmessage.dat	Data file that contains error, informational, and warning messages that are returned by the 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 GoldenGate.
ggsmsg.dll	Windows dynamic link library used by the INSTALL program.
GLOBALS	User-created file that stores parameters applying to the GoldenGate instance as a whole.
help.txt	Help file for the GGSCI command interface.
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 GoldenGate's XML procedures.
libxml2.txt	License agreement for libxml2.dll.
marker.hist	File created by Replicat if markers were passed from a NonStop source system.
marker_remove.sql	Script that removes the DDL marker table. (Oracle installations)
marker_setup.sql	Script that installs the GoldenGate DDL marker table. (Oracle installations)

Table 8 Other files (continued)

Component	Description
marker_status.sql	Script that confirms successful installation of the DDL marker table. (Oracle installations)
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.
role_setup.sql	Script that creates the database role necessary for 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.
zlib.txt	License agreement for zlib compression library.

GoldenGate checkpoint table

When database checkpoints are being used, 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 the type of database.

Do not change the names or attributes of the columns in this table. You can change table storage attributes as needed.

Table 9 Checkpoint table definitions

Column	Description
GROUP_NAME (primary key)	The name of a Replicat group using this table for checkpoints. There can be multiple Replicat groups using the same table.
GROUP_KEY (primary key)	A unique identifier that, together with GROUPNAME, uniquely identifies a checkpoint regardless of how many Replicat groups are writing to the same table.
SEQNO	The sequence number of the checkpoint file.
RBA	The relative byte address of the checkpoint in the file.
AUDIT_TS	The timestamp of the checkpoint position in the checkpoint file.
CREATE_TS	The date and time when the checkpoint table was created.
LAST_UPDATE_TS	The date and time when the checkpoint table was last updated.
CURRENT_DIR	The current GoldenGate home directory or folder.

Index

G

I

K

L

M

N

O

P

R

S

T

U

V

W