Teradata to Teradata: Max Protection

Objective

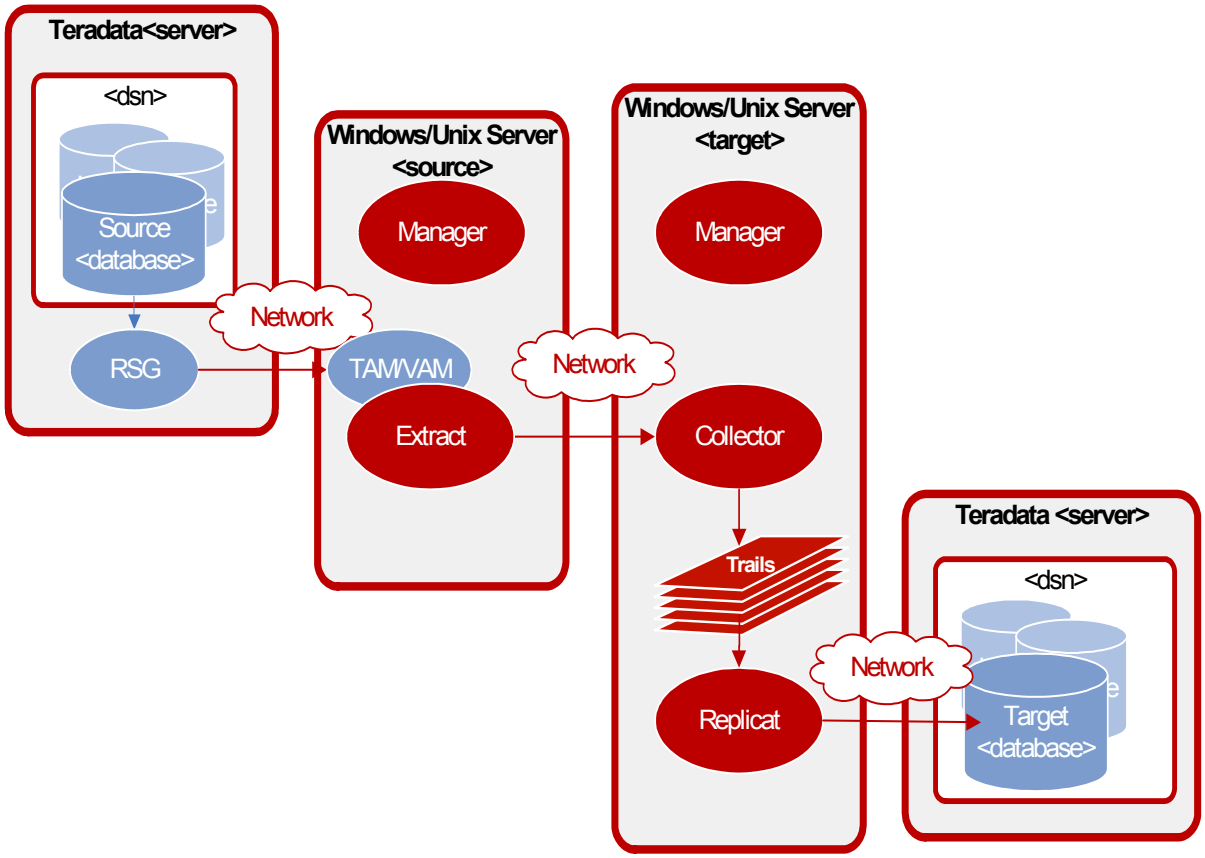
Upon completion of this lesson, you will be able to keep two Teradata databases synchronized using the maximum protection method.

During this lesson, you will learn how to:

- Prepare your user environment.
- Configure the change capture process to create a transaction log of Teradata changes in Max Protection mode.
- Execute the table copy process for the initial data synchronization process.
- Configure and start the change delivery process of database operations.

Teradata to Teradata max protection configuration

The diagram illustrates GoldenGate for Teradata in maximum protection mode.



GoldenGate is installed on a separate Windows or UNIX server for the source and target systems. Source and target both communicate with a Teradata server.

Overview of tasks

Prepare the Environment

In order to execute this lesson, the GoldenGate application must be installed on both the source and target systems. The installation includes a sample database and scripts to generate initial data as well as subsequent update operations. The source and target tables are created and loaded with initial data. The GoldenGate Manager processes are also started so that other processes may be configured and started.

Initial Data Load

To initially load data, you have a few strategies to select from, including Teradata's FastLoad utility or GoldenGate's TableCopy feature, both of which have their pros and cons.

This lesson demonstrates using the Teradata FastLoad utility to load the data at high speed to the target system.

Configure Change Capture

For VAM-based Teradata capture in Max Protection mode, the Extract process is configured to capture change data directly from the Teradata Access Module (TAM) and store the changes in data queues known as GoldenGate Trails.

In Max Protection mode, the capture stores uncommitted transactional detail to a VAM trail. Once the transaction is committed, GoldenGate persists the data and sends a message back to Teradata indicating that the transaction has been successfully saved. Teradata will then release the transaction. If the data cannot be persisted, Teradata will rollback the transaction. A VAM trail must then be processed by another Extract capture process to sort all of the transactional details in commit order.

In summary, two capture processes are configured. The first Extract creates a VAM log of all operations and the second Extract sorts them into commit order and creates a GoldenGate trail.

Configure Change Delivery

Once the tables have been initially loaded with data, the Replicat process is configured to deliver the captured change data into the target database.

Prerequisites

The prerequisites for this lab include the following.

- GoldenGate installed in both the source **<install location>** and the target **<install location>.**
- Teradata ODBC installed in the Windows environment.

- Teradata Change Data Capture facility of the Teradata Database must be installed and configured.
- Teradata Access Module library, `TAM.dll`, copied to the GoldenGate installation location.
- Students should have been granted the security necessary to create replication groups.

The Teradata ODBC, Change Data Capture facility, and the Teradata Access Module library are provided by NCR Corporation.

Exercise 1.
Prepare the Environment

Objective

The goals of this exercise are to:

- Configure and start the Manager processes
- Prepare source and target sample Teradata databases
- Create ODBC data source name
- Configure Teradata replication services
- Configure the TAM initialization file.

Configure the Manager processes

1. Configure the Manager process on the source

Execute the following commands on the **<source>** system.

- Start the command interface

```
shell> cd <install location>
shell> ggsci
```

- Specify the port that the Manager should use.

```
GGSCI> EDIT PARAMS MGR
```

```
-- GoldenGate Manager Parameter file
PORT <port>
```

- Start Manager

```
GGSCI> START MANAGER
```

- Verify the results:

```
GGSCI> INFO MANAGER
```

Prepare the database

2. Create ODBC data source name

For Unix/Linux the ODBC driver specification is defined in a text file named .odbc.ini. Note that the leading period in .odbc.ini is a required entry.

It is a prerequisite of this lab that the Teradata ODBC software packages are installed on your system. The typical location is /usr/odbc, but if your installation is in a different location substitute that location in the .odbc.ini[1].

To configure the Teradata ODBC driver for your database, create a text file called .odbc.ini in the home directory of the user executing the GoldenGate processes (for example, /home/gguser/.odbc.ini). A sample configuration is shown below. Note that:

- Lines beginning with "#" are comments.
- The first section, beginning with [ODBC], must define the installation directory of the Teradata ODBC driver.
- The second section, [ODBC Data Sources], lists any number of data source names (<dsn>s) that will be defined in the file.
- The remainder of the file is used to define the individual data sources declared under [ODBC Data Sources]. For example, there is a section called [<dsn>] below defining the connection settings to a Teradata database on host <server>. A special entry [default] can be used to list default data source settings for ODBC data sources not listed under [ODBC Data Sources] (not shown in the example).

```
# Teradata ODBC data source specifications
# Teradata ODBC install directory (required).
# Optionally specify trace settings.
[ODBC]
InstallDir=/usr/odbc

# List of data sources specifications defined in this file, along
# with their corresponding Teradata driver. If a requested data
# source is not listed, the special [default] entry will be used.
[ODBC Data Sources]
<dsn>=tdata.so

# The GGSTera ODBC driver settings. The driver path and the DBCName
# are required, other settings are optional.  See: man tdata(5)
[<dsn>]

Driver=/usr/odbc/drivers/tdata.so
Description=Generic ODBC to <server> v6
DBCName=<server>
```

[1] See the "Teradata ODBC for Unix Users Install Guide" for details for your version of Unix / Linux. For example, on Linux, using the RPM package manager, you can check for the existence of the Teradata ODBC packages by running: shell> rpm -aq -i tdodbc.

```
# Username/password to connect. If password expires, update this file
Username=<login>
Password=<password>

# Default database to use, if none specified.
DefaultDatabase=<database>
Database=<database>

# For GoldenGate, it's recommended to set the SessionMode
# to ANSI, unless directed otherwise.
SessionMode=ANSI

# Set DATE, TIME, and TIMESTAMP (respectively) to ANSI, not Integer
DateTimeFormat=AAA

# Driver should not display error messages on screen
RunInQuietMode=Yes

# Driver should not parse SQL; rather, send directly to Teradata
NoScan=Yes
```

Note: If this file needs to be located somewhere other than in the user's home directory, set the ODBCINI environmental variable to the absolute path of the file, for example:

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

3. Create source tables and load with initial data

Execute the following commands on the **<source>** system.

Edit the script, demo_tera _create.sql and demo_tera_insert.sql and change the **<database>** to the identifier for your source database.

```
create table <database>.tcustmer;
create table <database>.tcustord;
```

For the script demo_tera_insert.sql also enter the correct **<server>/<login>, <password>.**

Using the Teradata command interface, BTEQ, logon and execute the script.

```
Shell> bteq
BTEQ> .logon <server>/<login>
```

This will prompt you to enter the <password>. Then you can run the script.

```
BTEQ> .run file = ./demo_tera_create.sql;
BTEQ> .run file = ./demo_tera_insert.sql;
```

Verify the results for the tables used in this lab:

```
BTEQ> select * from <database>.tcustmer;
BTEQ> select * from <database>.tcustord;
```

5. Create the obey file to configure the Teradata replication group

On the **<source>** system, create the sql obey file **RepGroupProt.sql** with the following statement. Change the **<database>** and **<unique id>** to your values.

```
Create Replication Group REPGROUP<unique id> (<database>.tcustmer,
<database>.tcustord, <database>.statecode);
```

6. Configure the Teradata Access Module initialization file VAMPRO.ini

The initialization (.ini) file contains information to be passed to the TAM. These details include the replication mode, ODBC connection strings, the security token for the Replication group, and other TAM variables.

On your **<source>** system, use an editor such as WordPad to create a text file named <install location>\VAMPRO.ini. This should contain the following information set to your **<source>** values.

Note: As of Teradata 12.0 the **ReplicationGroupName** parameter is no longer required.

```
; Sets the mode for this extract process ("Replication" or "TableCopy")
Mode=Replication

; ODBC connection string used to access metadata from
; the data dictionary of the primary Teradata system
DictOdbcConnString=DSN=<dsn>;uid=<login>;pwd=<password>

; ODBC connection string used to execute management functions
; (such as CREATE REPLICATION GROUP) on the primary Teradata system
MgmtOdbcConnString=DSN=<dsn>;uid=<login>;pwd=<password>

; Name of the replication group is not needed when using
; CreateGroupStmtFile
; Otherwise for Teradata v2r6 enter (must be uppercase):
; ReplicationGroupName=REPGROUP<unique id>
; For Teradata v12 enter:
; GroupID= (enter the group ID number)

; If the group already exists, enter Security token
; SecurityToken= (enter the security token)
; OR enter the name of the file containing CREATE REPLICATION GROUP
; statement for a new group (set up in step 4)
CreateGroupStmtFile=./RepGroupPro.sql

; Character set (ASCII, UTF16)
CharacterSet=ASCII

; RSG Node addresses - specify name or IP, with optional port
ControlRSG=<server >:5298
DataRSG1=<server>:5298
; As of Teradata 12.0 multiple RSG connections may be included
```

```
; DataRSG2=<server>:<port>
; DataRSG3=<server>:<port>

; Specifies which types of messages are encrypted (None, Control, Data,
; or All)
Encryption=None
```

Prepare the Teradata target environment

1. Configure the Manager process

Execute the following command on the **<target>** Teradata system.

- Start the command interface

```
shell> cd <install location>
shell> ggsci
```

- Specify the port that the Manager should use.

```
GGSCI> EDIT PARAMS MGR
```

```
-- GoldenGate Manager Parameter file
PORT <port>
```

- Start Manager

```
GGSCI> START MANAGER
```

- Verify the results:

```
GGSCI> INFO MANAGER
```

2. Create ODBC data source name

For Unix/Linux the ODBC driver specification is defined in a text file named .odbc.ini. Note that the leading period in .odbc.ini is a required entry.

It is a prerequisite of this lab that the Teradata ODBC software packages are installed on your system. The typical location is /usr/odbc, but if your installation is in a different location substitute that location in the .odbc.ini[2].

To configure the Teradata ODBC driver for your database, create a text file called .odbc.ini in the home directory of the user executing the GoldenGate processes (for example, /home/gguser/.odbc.ini). A sample configuration is shown below. Note that:

[2] See the "Teradata ODBC for Unix Users Install Guide" for details for your version of Unix / Linux. For example, on Linux, using the RPM package manager, you can check for the existence of the Teradata ODBC packages by running: shell> rpm -aq -i tdodbc.

- Lines beginning with "#" are comments.
- The first section, beginning with **[ODBC]**, must define the installation directory of the Teradata ODBC driver.
- The second section, **[ODBC Data Sources]**, lists any number of data source names (**<dsn>**s) that will be defined in the file.
- The remainder of the file is used to define the individual data sources declared under **[ODBC Data Sources]**. For example, there is a section called [**<dsn>**] below defining the connection settings to a Teradata database on host **<server>**. A special entry **[default]** can be used to list default data source settings for ODBC data sources not listed under **[ODBC Data Sources]** (not shown in the example).

```
# Teradata ODBC data source specifications
# Teradata ODBC install directory (required).
# Optionally specify trace settings.
[ODBC]
InstallDir=/usr/odbc

# List of data sources specifications defined in this file, along
# with their corresponding Teradata driver. If a requested data
# source is not listed, the special [default] entry will be used.
[ODBC Data Sources]
<dsn>=tdata.so

# The GGSTera ODBC driver settings. The driver path and the DBCName
# are required, other settings are optional.  See: man tdata(5)
[<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=<login>
Password=<password>

# Default database to use, if none specified.
DefaultDatabase=<database>
Database=<database>

# For GoldenGate, it's recommended to set the SessionMode
# to Teradata, unless directed otherwise.
SessionMode=Teradata

# Set DATE, TIME, and TIMESTAMP (respectively) to ANSI, not Integer
DateTimeFormat=AAA

# Driver should not display error messages on screen
RunInQuietMode=Yes
```

```
# Driver should not parse SQL; rather, send directly to Teradata
NoScan=Yes
```

Note: If this file needs to be located somewhere other than in the user's home directory, set the ODBCINI environmental variable to the absolute path of the file, for example:

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

3. Create target files

Execute the following commands on the **<target>** system.

Edit the script, demo_tera_create.sql and change the **<database>** to the identifier for your target database.

```
create table <database>.tcustmer;
create table <database>.tcustord;
```

Using the Teradata command interface, BTEQ, logon and execute the script.

```
Shell> bteq
BTEQ> .logon <server>/<login>
```

This will prompt you to enter the <password>. Then you can run the script.

```
BTEQ> .run file = ./demo_tera_create.sql;
```

Exercise 2.
Initial Data Load

Objective

The goal of this exercise is to load the initial data from the source to the target using the Teradata FastExport and FastLoad.

Configure initial load capture

1. Execute the FastExport script on the source

Execute the following commands on the **<source>** system.

Edit the script, demo_tera_export.sql and change it to point to your source <database> and <install location>. Then logon to BTEQ to execute the script.

```
BTEQ> .logon <server>/<login>
         Password <password>

BTEQ> .run file = <install location>/demo_tera_export.sql;
```

2. Execute the FastLoad script on the target

Execute the following commands on the **<target>** system.

Edit the scripts, demo_tera_load_tcustmer.sql and demo_tera_load_tcustord.sql to point to your target <database> and <install location>. Enter your <server>, <login>, and <password>. Then execute the scripts using Teradata's FastLoad utility.

```
Shell> fastload < <install location>/demo_tera_load_tcustmer.sql;

Shell> fastload < <install location>/demo_tera_load_tcustord.sql;
```

Exercise 3.
Configure Change Capture

Objective

The goals of this exercise are to:

- Add an Extract process that will store uncommitted changes to a local VAM trail.
- Implement *max protection mode* by adding a second VAM-sort extract to read the VAM trail and sort the transactions into commit order in a remote GoldenGate trail.
- Start the capture process.

Configure primary change capture

1. Add the Extract group

Execute the following commands on the **<source>** system to create a capture (Extract) group named EPRO<unique id>.

```
GGSCI> ADD EXTRACT EPRO<unique id>, VAM
```

Verify the results:

```
GGSCI> INFO EXTRACT EPRO<unique id>
```

2. Create the Extract parameter file

Execute the following commands on the **<source>** system.

```
GGSCI> EDIT PARAM EPRO<unique id>
```

Note: The parameter IGNOREMETADATAFROMVAM is valid for Teradata v2r6, but not for 12, so you may need to alter the line that is commented out.

```
--
-- Change capture parameter file to capture
-- TCUSTMER and TCUSTORD changes
--
EXTRACT EPRO<unique id>
SOURCEDB <dsn>, USERID <login>, PASSWORD <password>
-- For Teradata v2r6
DSOPTIONS CREATETRANLOG IGNOREMETADATAFROMVAM
-- For Teradata 12
--DSOPTIONS CREATETRANLOG
WILDCARDRESOLVE DYNAMIC
VAM Tam.so, PARAMS(inifile, vampro.ini, callbackLib, extract)
EXTTRAIL ./dirdat/<trail id>
TABLE <database>.*;
```

Note: Record the two characters selected for your <trail id>: _____. You will need this in the next step and in the next section when you set up the VAM-sort Extract.

Verify the results:

```
GGSCI> VIEW PARAMS EPRO<unique id>
```

3. Define the GoldenGate trails

Execute the following on the **<source>** system.

```
GGSCI> ADD EXTTRAIL ./dirdat/<trail id>, EXTRACT EPRO<unique id>,
MEGABYTES 50
```

Verify the results:

```
GGSCI> INFO EXTTRAIL *
```

4. Start the primary capture process

Execute the following on the **<source>** system.

```
GGSCI> START EXTRACT EPRO<unique id>
```

Verify the results:

```
GGSCI> INFO EXTRACT EPRO<unique id>, DETAIL
GGSCI> VIEW REPORT EPRO<unique id>
```

Configure VAM-sort data pump

5. Add the VAM-sort Extract group

Execute the following commands on the **<source>** system.

```
GGSCI> ADD EXTRACT ESRT<unique id>, VAMTRAILSOURCE ./dirdat/<trail
id>
```

Note: This is the two character <trail id> added with the primary Extract.

Verify the results:

```
GGSCI> INFO EXTRACT ESRT<unique id>
```

6. Create the VAM-sort Extract parameter file

Execute the following commands on the **<source>** system.

```
GGSCI> EDIT PARAM ESRT<unique id>
```

```
--
-- Change Capture parameter file to capture
-- TCUSTMER and TCUSTORD Changes
--
EXTRACT ESRT<unique id>
SOURCEDB <dsn>, USERID <login>, PASSWORD <password>
DSOPTIONS SORTTRANLOG
WILDCARDRESOLVE DYNAMIC
RMTHOST <target>, MGRPORT <port>

RMTTRAIL ./dirdat/<trail id>
TABLE <database>.*;
```

Note: Record the two characters selected for your second <trail id>: _____. You will need this in the next step and later when you set up the Replicat.

Verify the results:

```
GGSCI> VIEW PARAMS ESRT<unique id>
```

7. Define the GoldenGate remote trails

Execute the following commands on the **<source>** system.

```
GGSCI> ADD RMTTRAIL ./dirdat/<trail id>, EXTRACT ESRT<unique id>,
MEGABYTES 50
```

Verify the results:

```
GGSCI> INFO RMTTRAIL ./dirdat/<trail id>
```

8. Start the VAM-sort Extract process

```
GGSCI> START EXTRACT ESRT<unique id>
```

Verify the results:

```
GGSCI> INFO EXTRACT ESRT<unique id>, DETAIL
GGSCI> VIEW REPORT ESRT<unique id>
```

Discussion points

1. Using max protection mode

Why are two Extracts used with max protection mode?

_____

_____

2. The role of DSOPTIONS

Search for DSOPTIONS in the *Oracle GoldenGate Reference* for help answering the following questions.

Does your lab use the **COMMITTEDTRANLOG** or **CREATETRANLOG** option? Why that one and not the other?

_____

_____

Does your lab use **SORTTRANLOG**? Where is it used and why?

_____

_____

Exercise 4.
Configure Change Delivery

Objective

The goals of this exercise are to:

- Set up the checkpoint table on the target system.
- Create a named group that includes the Replicat process and the checkpoint tables.
- Configure the Replicat group by adding parameters.
- Start the Replicat group.

Set up the checkpoint table

1. Create a GLOBALS file on the target system

Execute the following commands on the **<target>** system.

- Create and edit the **GLOBALS** parameter file to add the checkpoint table.

```
Shell> cd <install location>
Shell> ggsci
GGSCI> EDIT PARAMS ./GLOBALS
```

In the text editor, type:

```
CHECKPOINTTABLE <database>.ggschkpt
```

- Record the checkpoint table owner and name, then save and close the file.

Table owner _____ name _____

Note: You could name the table anything you want, but for training purposes we are using **ggschkpt**.

- Verify that the **GLOBALS** file was created in the root GoldenGate directory, and remove any file extension that was added.

2. Activate the GLOBALS parameters

For the **GLOBALS** configuration to take effect, you must exit the session in which the changes were made. Execute the following command to exit **GGSCI**.

```
GGSCI> EXIT
```

3. Add a Replicat checkpoint table

On the **<target>** system, execute the following commands in **GGSCI**:

```
Shell> cd <install location>
Shell> ggsci
GGSCI> DBLOGIN SOURCEDB <dsn>, USERID <login>, PASSWORD <password>
GGSCI> ADD CHECKPOINTTABLE
```

Configure Change Delivery

4. Add the Replicat group

Execute the following command on the **<target>** system to add a delivery group named RTER<unique id>.

```
GGSCI> ADD REPLICAT RTER<unique id>, EXTTRAIL ./dirdat/<trail id>
```

Note: Refer to your Extract set up for the correct two-character <trail id>.

5. Create Replicat parameter file

Execute the following commands on the **<target>** system to bring up the parameter file in the editor.

```
GGSCI> EDIT PARAM RTER<unique id>
```

Type in the following parameters

```
--
-- Change Delivery parameter file to apply
-- TCUSTMER and TCUSTORD Changes
--
REPLICAT RTER<unique id>
TARGETDB <dsn>, USERID <login>, PASSWORD <password>
HANDLECOLLISIONS
ASSUMETARGETDEFS
DISCARDFILE ./dirrpt/RTER<unique id>.DSC, PURGE
MAP <database>.tcustmer, TARGET <database>.tcustmer;
MAP <database>.tcustord, TARGET <database>.tcustord;
```

6. Start the Replicat process

```
GGSCI> START REPLICAT RTER<unique id>
```

Verify the results:

```
GGSCI> INFO REPLICAT RTER<unique id>
```

Discussion points

1. When to use HANDLECOLLISIONS

When would you use HANDLECOLLISIONS? What does it do?

_____

_____

2. When to use ASSUMETARGETDEFS

What should be the same on the source and target when ASSUMETARGETDEFS is used?

_____

_____

3. What is the purpose of the DISCARDFILE?

_____

_____

Exercise 1.
Generate Activity and Verify Results

Objective

The goals of this exercise are to:

- Execute miscellaneous update, insert, and delete operations on the source system.
- Verify the delivery of the changes to the target.
- Turn off the error handling used for the initial load.

Generate database operations

1. Execute miscellaneous update, insert, and delete operations

Execute the following commands on the **<source>** system.

Edit the script, **demo_tera_misc.sql** and change the **<database>** to be your source database identifier. Then logon to BTEQ and run the script to generate the operations.

```
BTEQ> .logon <server>/<login>
         Password <password>

BTEQ> .run file = ./demo_tera_misc.sql;
```

Verify change capture and delivery

2. Verify your results on the source system

Execute the following commands on the **<source>** system.

```
BTEQ> select * from <database>.tcustmer;
BTEQ> select * from <database>.tcustord;

Shell> cd <install location>
Shell> ggsci
GGSCI> SEND EXTRACT ETER<unique id>, REPORT
GGSCI> VIEW REPORT ETER<unique id>
```

3. Verify your results on the target system

Execute the following commands to logon to BTEQ on the **<target>** system and verify the target data.

```
Shell> bteq
BTEQ> .logon <server>/<login>
```

This will prompt you for the <password>. After that is entered you may query the database.

```
BTEQ> select * from <database>.tcustmer;
BTEQ> select * from <database>.tcustord;

Shell> cd <install location>
Shell> ggsci
GGSCI> SEND REPLICAT RTER<unique id>, REPORT
GGSCI> VIEW REPORT RTER<unique id>
```

Turn off error handling

4. Turn off initial load error handling for the running delivery process

```
GGSCI> SEND REPLICAT RTER<unique id>, NOHANDLECOLLISIONS
```

5. Remove initial load error handling from the parameter file

```
GGSCI> EDIT PARAMS RTER<unique id>
```

Remove the HANDLECOLLISIONS parameter.

