Oracle® GoldenGate

Administering Oracle GoldenGate for HP NonStop (Guardian Data Sources)

26ai
G52380-01
March 2026

Oracle GoldenGate Administering Oracle GoldenGate for HP NonStop (Guardian Data Sources), 26ai

G52380-01

Contents

2 Planning the Configuration

3 Configuring Initial Data Synchronization

4 Configuring Oracle GoldenGate Security

5 Configuring the Manager and Collector

6 Configuring Change Synchronization

7 Configuring Custom Operations

8 Integrating Data

9 Managing and Monitoring

10 Using Oracle GoldenGate Utilities

A Oracle Oracle GoldenGate Components

B Installing Event Detail Text

Preface

This guide contains information on configuring, and running Oracle GoldenGate on the HP NonStop Guardian platform for Enscribe and SQL/MP. This product does not support SQL/MX.

Audience

This guide is intended for system administrators who are configuring, and running Oracle GoldenGate on the HP NonStop Guardian platform.

Documentation Accessibility

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

Accessible Access to Oracle Support

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

Related Information

Oracle GoldenGate Documentation

Oracle GoldenGate for Distributed Applications and Analytics

OCI GoldenGate

Oracle Database High Availability

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, such as "From the File menu, select **Save**." Boldface also is used for terms defined in text or in the glossary.
italic *italic*	Italic type indicates placeholder variables for which you supply particular values, such as in the parameter statement: TABLE *table_name*. Italic type also is used for book titles and emphasis.

Convention	Meaning
`monospace` `MONOSPACE`	Monospace type indicates code components such as user exits and scripts; the names of files and database objects; URL paths; and input and output text that appears on the screen. Uppercase monospace type is generally used to represent the names of Oracle GoldenGate parameters, commands, and user-configurable functions, as well as SQL commands and keywords.
UPPERCASE	Uppercase in the regular text font indicates the name of a utility unless the name is intended to be a specific case.
`{}`	Braces within syntax enclose a set of options that are separated by pipe symbols, one of which must be selected, for example: `{`*`option1`* `|` *`option2`* `|` *`option3`*`}`.
`[]`	Brackets within syntax indicate an optional element. For example in this syntax, the SAVE clause is optional: `CLEANUP REPLICAT` *`group_name`* `[,` `SAVE` *`count`*`]`. Multiple options within an optional element are separated by a pipe symbol, for example: `[`*`option1`* `|` *`option2`*`]`.

1

Understanding Oracle GoldenGate for HP NonStop

This topic describes the capabilities of Oracle GoldenGate for HP NonStop to manipulate data at the transactions level and to replicate selected data to a variety of heterogeneous applications and platforms. It introduces both the configuration and the features of Oracle GoldenGate for HP NonStop.

This topic includes the following sections:

Oracle GoldenGate Overview

Oracle GoldenGate for HP NonStop has modular architecture that gives you the flexibility to extract and replicate selected data records and transactional changes across a variety of heterogeneous applications and platforms.

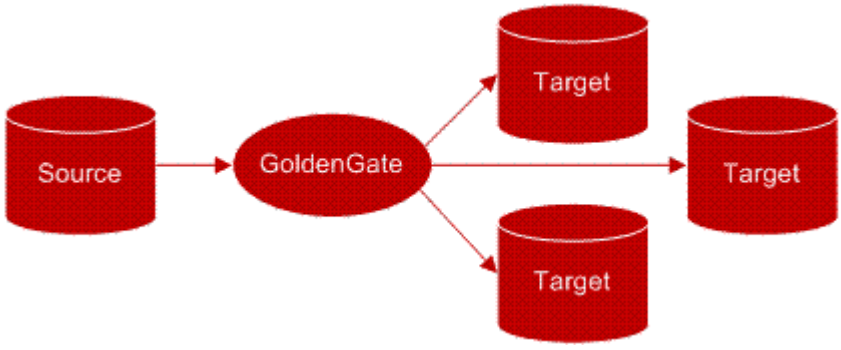
You can configure Oracle GoldenGate for HP NonStop to manage data from multiple, heterogeneous sources and targets. Oracle GoldenGate for HP NonStop contains features that enables your business to manage data at the transaction level across the enterprise.

Oracle GoldenGate Configuration

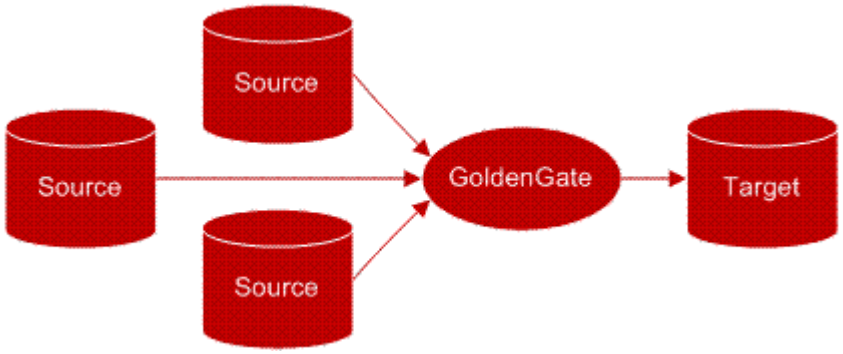
Oracle GoldenGate offers flexibility in configuring your transaction management system, supporting both homogeneous and heterogeneous data replication. This lets you configure Oracle GoldenGate to capture and deliver data to best suit your operating environment. Options include:

1. One-to-one, from a single source to a single target

2. One-to-many, from a single source to multiple targets



3. Many-to-one, from multiple sources to a single target



4. Bi-directional, between a single source and a single target

In doing so, the following business needs can be met:

- Change synchronization, supporting both online and batch change synchronization for Transaction Management Facility (TMF)-enabled and non-TMF-enabled applications.
  - Online change synchronization continuously processes incremental data changes.
  - Batch change synchronization processes change records that are generated during specific periods of time.
- Initial load, extracting complete records directly from a source file or table, then loading them into the target. You can use initial load to populate the target and to synchronize the source and target for change synchronization.
- Data distribution, sending extracted records to more than one target.

Oracle GoldenGate Features

Oracle GoldenGate features let you select, map, and transform data so it can be used for a variety of applications. You can configure Oracle GoldenGate to integrate and convert data by selecting data based on filtering criteria. You can implement custom logic so Oracle GoldenGate works seamlessly with your own applications.

For example, you can configure the activities of Oracle GoldenGate by:

- Configuring data selection to deliver only required records, filter records to extract specific column data, and control which types of operations are extracted
- Mapping named source files and tables to the target when the target has similar formats but different file or table names
- Splitting single rows into multiple rows and combining rows from different tables to a single table
- Implementing data transformations to:
  - Convert dates from one format to another

  - Perform arithmetic calculations
  - Transform DML operations, such as converting delete operations into insert operations on the target table

You can also configure Oracle GoldenGate to run your custom programs and frequently-used routines with user exits, macros, and obey files. These features increase the flexibility of Oracle GoldenGate by:

- Inserting user exits to call your applications and/or custom data management logic
- Using macros to implement multiple uses of a statement, consolidate multiple commands, and call other macros
- Automating frequently-used routines by using the `OBEY` command, which instructs Oracle GoldenGate to process parameters specified in another parameter file.

The modular architecture of Oracle GoldenGate lets you implement just the components you need. These components are introduced in the next section.

Oracle GoldenGate Architecture

Oracle GoldenGate processes data by capturing records from a data source, housing it temporarily, then delivering it to a data target. Each of these steps is handled by a modular component of Oracle GoldenGate.

Oracle GoldenGate Components

Oracle GoldenGate for HP NonStop consists of the following components:

- Extract, for extracting and processing records from Transaction Monitoring Facility (TMF)-enabled applications
- Logger, for extracting and processing records from non-TMF-enabled applications
- Collector, for facilitating the transmittal of records between local and remote systems
- Oracle GoldenGate trails, for transmitting change records from the source to the target
- Replicat, for processing and replicating records to a target
- Oracle GoldenGate Manager, for controlling, monitoring, and reporting on Oracle GoldenGate processing.
- Checkpoint groups, for helping maintain data integrity by tracking where, on the source, processing starts and stops
- Parameters, for compiling instructions for Extract, Replicat, Manager, and utilities
- Reader, for monitoring Oracle GoldenGate trails for distributed network transactions and communicating status information to the Coordinator
- Coordinator, for tracking distributed network transactions to coordinate processing across multiple nodes

Extract

Extract extracts source data from the TMF audit trail and writes it to one or more files, called Oracle GoldenGate trails. Multiple Extracts can operate on different sources at the same time. For example, one Extract could continuously extract data changes from a database and stream them to an up-to-date decision-support database, while another Extract performs batch extracts from other tables for periodic reporting. Or, two Extract processes can extract and

transmit in parallel to two Replicat processes to minimize target latency when the databases are large.

Use Extract for:

- Initial load
- Change synchronization for TMF-enabled applications
- Transmitting change records between Logger (non-TMF) over TCP/IP to a remote target
- Data distribution

Logger

Logger performs data extracts when a NonStop source is non-TMF. Logger requires GGSLIB, an intercept library, that binds the Oracle GoldenGate application to the user's NonStop application. When the application performs an Enscribe operation (such as `WRITE`), GGSLIB intercepts it and sends the record to Logger. Logger writes the records to a log trail which is read by Replicat.

Collector

When data is transmitted over a TCP/IP connection, the Collector resides on the target system and receives incoming records. Each Replicat group has a dedicated Collector process that terminates when the group's Extract process terminates.

Trails

Extract and Logger create trails to transmit data from the source to the target. An Oracle GoldenGate trail can contain a sequence of files or a single flat file. Generally, an Oracle GoldenGate trail is used during online change synchronization and an Oracle GoldenGate file is used for one-time tasks, such as initial data load or certain batch processes.

All trail file names begin with the same two characters, which you specify when you create the trail. As files are created, the name is appended with a six-digit number that increments sequentially from 000000 to 999999. When the sequence number reaches 999999, the numbering starts over at 000000.

There are two kinds of Oracle GoldenGate trails:

- Local trails. Local trails are transmitted to another NonStop system over Expand and read by Replicat. Local trails can also reside on the source and be used as a data source for Extract.
- Remote trails. Remote trails are transmitted to the target over TCP/IP and read by Replicat on the remote target.

Each record in an Oracle GoldenGate trail includes the data, a header with transaction information, an identifier of the record source, and other items. Trails are unstructured for best performance and are collected into transactions, which process in a continuous stream. Transaction identifiers indicate the first and last records in each transaction. Transactions are written in commit order, which guarantees that:

- Each record has been committed in the original database
- All records in the original transaction are together in the output
- Inserts, updates, and deletes are presented, per record key, in the order in which they were applied

To maximize throughput and minimize I/O load, extracted data is written to, and read from, the trail in large blocks. By default, Oracle GoldenGate writes data to the trail in a proprietary format which allows data to be exchanged rapidly and accurately among heterogeneous databases. However, data can also be written in external ASCII, XML, or other formats compatible with different applications.

Enabling Trail Recovery (FAR)

By default, Extract operates in an append mode. If there is a process failure, then a recovery marker is written to the trail and the Extract appends recovery data to the file. This is done to retain a history of all prior data recovery purposes. In an append mode, the Extract initialization determines the identity of the last complete transaction that was written to the trail at startup time.

With that information, Extract ends recovery when the commit record for that transaction is encountered in the data source; then it begins a new data capture with the next committed transaction that qualifies for extraction and then begins appending the new data to the trail. A data pump or Replicat starts reading again from that recovery point.

Overwrite mode is another version of Extract recovery that was used in versions of Oracle GoldenGate prior to version 10.0. In these versions, Extract overwrites the existing transaction data in the trail after the last write-checkpoint position, instead of appending the new data. The first transaction that is written is the first one that qualifies for extraction after the last read checkpoint position in the data source. This behavior can be controlled manually with the `FORMAT RELEASE option on EXTTRAIL/RMTTRAIL` parameter.

Replicat

Replicat reads data from Oracle GoldenGate trails that were created by Extract or Logger. You can run multiple instances of Replicat to read multiple Oracle GoldenGate trails. Replicat supports a high volume of replication activity on the target platform, transferring data in blocks rather than a single record at a time. SQL operations are compiled once and performed many times, and small transactions can be grouped into larger transactions to improve performance.

Manager

Oracle GoldenGate is managed by the Manager. Manager is responsible for starting and stopping Extract, Replicat, and their dependent subprocesses, such as Collector. Extract and Replicat checkpoints give Manager information about what resources are required at a particular time. No other Oracle GoldenGate processes will run if Manager is stopped.

Syncfile

Syncfile lets you schedule and manage file duplication when you want to copy files in their entirety. This is a common requirement for maintaining a secondary Oracle GoldenGate instance that may see frequent database changes, but infrequent configuration file changes. However, Syncfile can copy any type of file, database or not, according to a schedule set by you. This makes it suitable for other off-hours, small scale copying tasks.

Processing Groups

To differentiate among multiple Extract or Replicat processes on a system, you define processing groups. A processing group consists of a process (either Extract, Replicat, or Syncfile), its parameter file, its checkpoint file (if applicable), and any other files associated with the process. For example, to replicate two sets of data in parallel, you would create two

Replicat groups. You might name one group `GGSDEL1` and the other `GGSDEL2`. Groups are defined by the `ADD EXTRACT` and `ADD REPLICAT` commands in GGSCI.

A group name can contain up to eight characters and is not case-sensitive. All files and checkpoints relating to a group share that name. Any time you issue a command to control or view processing, you supply a group name or multiple group names with a wildcard.

You can use numbers in group names, but it is best to avoid placing numbers at the end. Oracle GoldenGate appends a numeric value of 0 to 9 to group names to create report file names. In an instance with Replicats `REP1` and `REP11`, for example, a report file will be created for `REP1` with the name `REP11`. This can cause confusion.

Checkpoints

Checkpoints are used to store the current read and write position of a process. They ensure that data changes marked for synchronization are extracted, and they prevent redundant extracts. Checkpoints provide fault tolerance by preventing the loss of data if the system, the network, or an Oracle GoldenGate process must be restarted. For advanced synchronization configurations, checkpoints enable multiple Extract or Replicat processes to read from the same set of trails.

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

The Extract process checkpoints its position in the data source and in the trail. The Replicat process checkpoints its position in the trail. The checkpoint position is a combination of the sequence number of the trail file and the Relative Byte Address (RBA) of the trail.

The read checkpoint is always synchronized with the write checkpoint. Thus, if Oracle GoldenGate must re-read data that was already sent to the target system (for example, after a process failure), checkpoints enable accurate overwriting of the old data to the point where new transactions start and Oracle GoldenGate can resume processing.

Parameters

Parameters manage all Oracle GoldenGate components and utilities, allowing you to customize your data management environment to suit your needs. For example:

- The Manager parameter file contains instructions for controlling all other Oracle GoldenGate processes.
- The Logger parameter file contains instructions for capturing data from non-TMF applications.
- The Extract parameter file contains instructions for selecting, mapping, and transforming TMF data, and sending trails to Replicat.
- The Replicat parameter file contains instructions for selecting, mapping, and transforming data to the target.
- The Global parameter file contains instructions that can be applied globally to Oracle GoldenGate processing.

Reader

A Reader on each node scans the local Oracle GoldenGate trail for distributed transactions. When one is found, the Reader gathers local transaction information and sends it to the Coordinator process.

Coordinator

The Coordinator process receives information from each replicating node on the status of the distributed network transactions that are being processed. The transaction is not committed until the Coordinator has been notified that all of the updates have been received on their destination nodes. If any node has a failure, the changes are not applied.

Oracle GoldenGate Processing

Oracle GoldenGate for NonStop processes data in a variety of ways, depending on your organization's needs and operating environment. This section introduces the primary ways Oracle GoldenGate captures and delivers data, including:

Initial Data Synchronization

Run initial data synchronization to synchronize the source and target databases. This process can be run while your transaction system is operational because Oracle GoldenGate will not lock data when it captures and delivers records. With Oracle GoldenGate, your options for loading data include:

- Extracting data to a file and sending it to Replicat to apply to the target
- Using Oracle GoldenGate direct load
- Using Oracle GoldenGate direct bulk load when the target is Oracle

File to Replicat

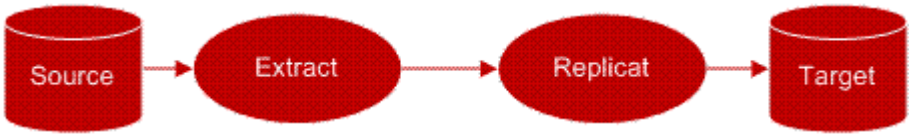
You can queue your data in one, or many, Oracle GoldenGate files before loading your target for the first time. This lets you perform initial data synchronization while your transaction system remains online.

Figure 1-1 File to Replicat Processing Flow

Direct Load

Using Oracle GoldenGate direct load lets you extract data directly from source tables and send it in large blocks directly to Replicat, which writes data to its final target. This method is particularly effective for source data that does not require transformation (such as initial data loads).

Figure 1-2 Direct Load Processing Flow



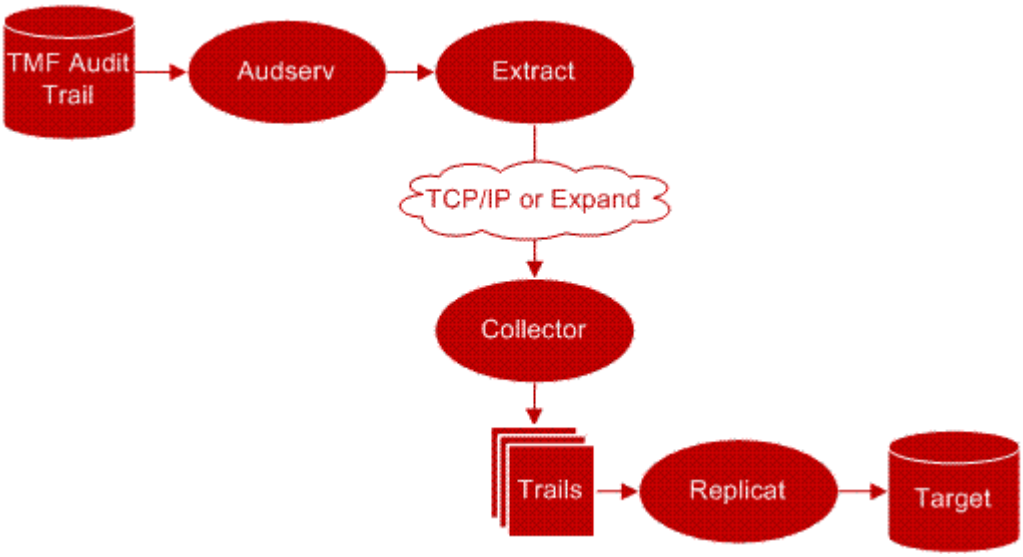
Capturing Data Changes from TMF Applications

TMF audit trails provide the central resource for retrieving database changes in TMF-enabled applications. Changes to TMF-enabled Enscribe files and SQL tables are recorded in TMF audit trails for transaction integrity and recoverability. The following figure shows the processing flow for TMF-audited applications.

> **Note**
>
> Because Oracle GoldenGate uses these audit trails for extract processing, plan and manage TMF-related activities carefully. The Oracle GoldenGate GGSCI and Manager tools provide optional audit management capabilities.

The Extract and Audserv work together to retrieve and process database changes for TMF applications. When started, Extract starts an Audserv process, which returns database changes from TMF audit trails. Audserv reads audit trails from their original location on disk, from a disk or tape dump, or from a user-specified alternative location. Audserv also determines the location of all required audit. Audserv can only areturn changes to tables or files if the user has read access.

Figure 1-3 TMF Audited Process Flow



Database changes include insert, update or delete operations, along with transaction information. Insert and update records are *after-images*, or the format of the database record after the operation completes; delete records returned are *before-images*. Before-images for updates can also be returned.

Extract saves each image in memory until an associated transaction commit record is received. If the transaction aborts, the associated records are discarded. Committed records can be sent to one or more user-designated extract files.

Audserv automatically excludes audit associated with SQL/MP and SQL/MX catalogs (file codes 564, 564, 565, 572, and 585).

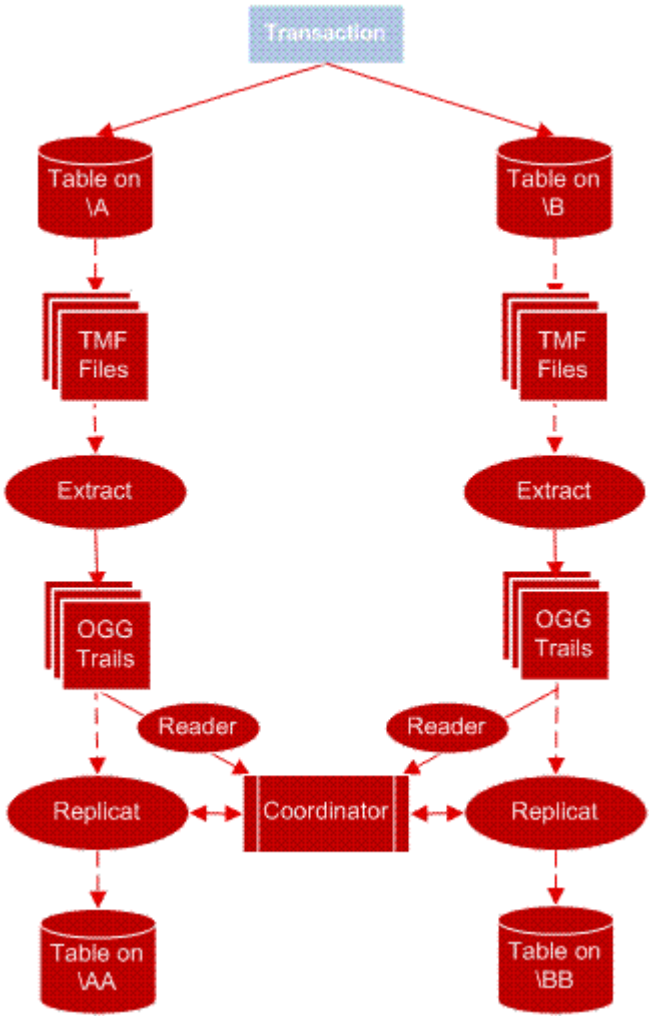
Capturing Changes for Distributed Network Transactions

In a multi-node environment a single transaction can update files across different nodes. Oracle GoldenGate includes processes to coordinate these network transactions so an outage on one of the nodes will not result in a partially updated transaction.

The central process is called the Coordinator. It receives information from each replicating node on the status of the changes being processed. The transaction is not committed until the Coordinator has been notified that all of the updates have been received on their destination nodes. If any node has a failure, the changes are not applied.

Figure 1-4 illustrates the processing flow for a transaction across two nodes. An order from a customer, for example, can add information to the customer account file on the `\A` and the customer order file on `\B`, and these files can each be replicated to backup nodes.

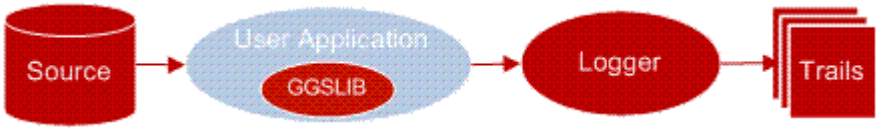
Figure 1-4 Processing Flow for Distributed Network Transactions



Capturing Data Changes from Non-TMF applications

Many Enscribe applications do not use the NonStop TMF audit facility. Oracle GoldenGate provides an alternative method for capturing non-TMF-audited database changes. Figure 5 displays the processing flow.

Figure 1-5 Oracle GoldenGate Processing Flow—Non-TMF Applications



The Oracle GoldenGate Software intercept library (GGSLIB) is a group of functions with the same names as Guardian operating system procedures. GGSLIB is bound to Guardian, acting as an interface between application programs and NonStop.

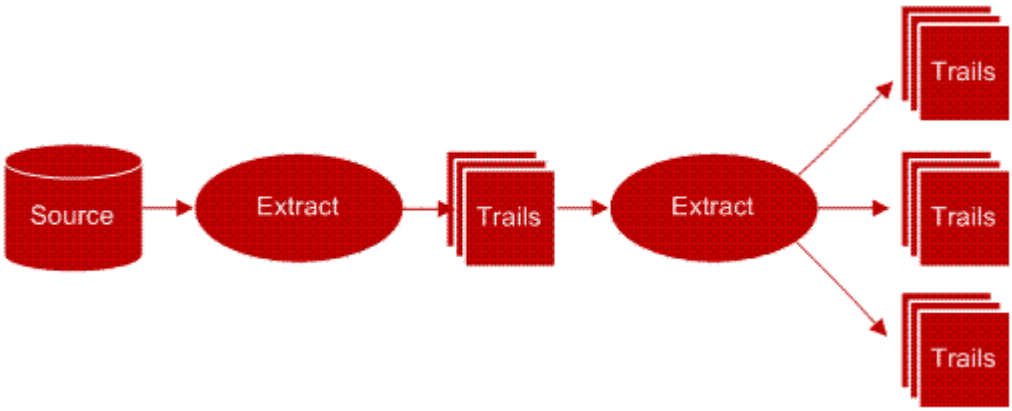
For example, when an application calls a Guardian function such as `WRITE`, GGSLIB performs it instead of Guardian. The application is unaware of the substitution and performs, from an application programming standpoint, exactly as it did before.

If the function succeeds, it sends its data to Logger, which writes it to an Oracle GoldenGate log trail. Log trails are available for Extract and Replicat processing, which perform formatting, distribution, and delivery steps.

Using Extract for Data Distribution

Extract can retrieve data from custom-created files or from trails created by Extract or Logger —in this sense, distributing data. Applications can take advantage of the data movement, formatting, conversion, and other features of Oracle GoldenGate without reading the data directly from TMF audit trails or the database.

Figure 1-6 Extract as Data Distributor



Batch Processing

When capturing incremental data changes in real-time is not appropriate, you can run batch processing. Batch runs process data generated during a specific time frame, defined by a

begin and end time. Many Oracle GoldenGate operations, such as record selection, mapping, and field conversions can be performed during batch processing.

Capturing Directly from Files

In some situations, Extract can read directly from a file rather than from the log trail; however, the following conditions must be true:

- The file or sequence of files is entry-sequenced.
- Only inserts occur against the files—no updates or deletes.
- Records are inserted only at the *end of the file.*

Use this feature when:

- The files are `Entry`, `BASE24 TFL/PTLF`, or Connex Advantage.
- The input files meet the conditions described above.
- You want to "trickle" the batch file contents throughout the day, rather than all at once at the end of the day.

Custom Event Processing

Oracle GoldenGate user exits make it possible to incorporate custom processing needs. A common application of user exits is database event triggering. For example, a user exit might contain code to page a supervisor when an account balance falls below a certain threshold. User exits reside outside the mainstream application—you can add, change, or remove them with almost no impact on the application.

Oracle GoldenGate Commands

GGSCI is the command-line interface that lets you interface with all Oracle GoldenGate components. Throughout this guide, you will see GGSCI commands described; they are your primary tools for configuring, operating, managing, and troubleshooting your data management environment.

Output from the Oracle GoldenGate GGSCI interface supports up to 1024 processing groups, including Extract, Coordinator, Syncfile, and Replicat groups. At the supported level, all groups can be controlled and viewed in full with the GGSCI commands, such as `INFO` and `STATUS` commands. Beyond that supported level, group information is not displayed and errors can occur. Oracle GoldenGate recommends keeping the combined number of processing groups at 1024 or below in order to manage your environment effectively.

To Start GGSCI

Before you can start GGSCI, you must navigate to the Oracle GoldenGate installation location. When you are in the correct subvolume, enter `RUN GGSCI`. Your prompt will change to a GGSCI prompt. For example:

Example 1-1 Starting GGSCI

```
TACL> VOLUME $DATA.GGS
TACL> RUN GGSCI
GGSCI>
```

After you have the GGSCI command prompt, enter GGSCI commands on the command line as needed. With GGSCI commands, you can edit parameter files, add groups, view reports, and communicate with running processes. For more information, see GGSCI Commands .

2

Planning the Configuration

Plan how to configure your Oracle GoldenGate for HP to suit your own business requirements. This topic outlines techniques and procedures for the following configuration tasks.

This topic includes the following sections:

Planning Overview

Before running Oracle GoldenGate, you must make some decisions regarding your Oracle GoldenGate installation. This includes determining your required resources and their configuration. Other planning considerations include:

- Are you capturing change data from TMF-enabled or non-TMF-enabled applications?
- Are you transmitting data to targets over TCP/IP?
- Which topology configurations are you using?
- How much data communications capacity is required between the source and target systems?
- How much additional disk space is required to accommodate replication?
- Can Oracle GoldenGate accommodate present and future transaction volumes?
- How much overhead will Oracle GoldenGate add to the source and target systems?
- How can you scale Oracle GoldenGate to meet high volume requirements?

Configuring TMF-Enabled Processing

Extract and its servant program Audserv read TMF data in large blocks (upwards of 28K at a time) before transferring the data to an Oracle GoldenGate trail. This requires a small percentage of I/O messages because Audserv retrieves blocks of records from audit cache rather than from disk.

You must consider several factors when planning for Audserv to read TMF data. These include:

Changing the Layout of a File or Table

Occasionally column layouts are changed to add or delete columns for source files and tables. This means if you need to change a layout of a file or table following some recommended procedures will help to avoid having any impact on Oracle GoldenGate processes.

SQL/MP tables that are altered to add columns are automatically extracted. To replicate these changes to a target MP table only requires that Replicat uses `REPNEWCOLUMNS`. Otherwise alter the target, then the source.

MP tables that are dropped and re-created only requires the Target node be updated first, then the source. If the tables are currently open by Extract or Replicat, you can issue a `CLOSEFILES` command from GGSCI. Table Create and Drop is not supported.

For changes to Enscribe DDL records in a dictionary, the dictionary should be updated first on both systems for its corresponding record definition. If the file is to be purged and re-created, and the end to end Extract(s) and Replicat are all using `GETFILEOPS`, nothing more is required. If the physical file is not changing and only the DDL, then stop/start the Extract(s) and Replicat.

> **Note**
>
> - Best practice is to complete file and table maintenance is while little to no DML is taking place.
> - If any parameter file(s) contains specific detailed column mapping might need additional changes depending on what has changed with the new layout and the target, especially if the target is on OpenSys.
> - If your Extract is using `FORMAT RELEASE` to exclude metadata in trails, then you must stop the Extract(s) and Replicat, complete file and table maintenance, Run `DEFGEN` (with the same `FORMAT RELEASE` <*nn*> as your Extract), transfer the new `SOURCEDEFS` file to the target, and restart the processes.

Ensuring All Audit is Processed

Various system events can require that you ensure all audit records are processed before the event occurs. Examples include system maintenance, (such as an operating system upgrade), TMF shutdown, and other events. Failing to do so can result in missed data.

There are several methods for verifying that Extract is current with TMF activity.

- Use the GGSCI `SEND EXTRACT AUDITEND` command to determine Extract's position in the audit trail. If the response indicates that all audit is processed, Extract has no more work to do assuming that TMF-related applications are down or idle.
- Use the GGSCI `ADD MARKER` command to insert a marker record into the audit trails after some significant event (such as taking the application down). Once Extract and Replicat have processed the marker, you can assume that all records before that point have been processed.
- Issue the `INFO EXTRACT` command from GGSCI, which returns the Extract lag (approximate number of bytes and time behind the audit trails). If the status is `RUNNING` and the number of bytes behind is less than 5000, it is likely that all audit has been processed.
- Issue a `LAG EXTRACT` command from GGSCI which reports the current lag times.

Keeping Necessary Audit Available for Extract

TMF purges audit trails it no longer requires, because it has no knowledge of outside processes that depend on it, such as Extract. This means you must plan how to keep audit trails available. This section discusses several options:

- Configure the audit trails to make disk dumps, and let Extract read them.
- Configure the audit trails to make tape dumps, and let Extract restore the audit.
- Include the `DISKTHRESHOLD` parameter in the Manager parameter file, so Manager warns you when audit trails are in danger of being purged.

Using Tape Dumps as an Alternative Location

If you specify a tape as the alternative location, Extract displays a message asking the operator to restore the tape. The Extract program restores tape dumps to one of two locations before processing the audit. In order of preference, the locations are:

- The first restore volume configured for the audit trail with TMFCOM
- The original location of the file

Restoring tape dumps before run time can be convenient. To determine which tapes must be restored for a specific Extract group, use the GGSCI `STATUS EXTRACT` command. The command lists the names of required audit files and whether they exist on disk or tape. All files on tape must be restored. The GGSCI `STATUS AUDITTRAIL` command lists the names of all audit trails required across all Extract groups.

Minimizing Vulnerability to Outages

Extended network or target system outages can have an adverse impact on Extract processing. When the intended target system is unavailable, Extract cannot process the audit trail. If the target system remains down, critical audit will eventually be deleted from the system before it can be processed.

To prevent this problem, extract the data to a local trail for Replicat to access over Expand. This solution only applies when both the source and target are NonStop systems.

An alternative is to extract the data from the audit trails to an intermediate Oracle GoldenGate trail on the source, then configure a second Extract to move data to the target system. This ensures that data can always be extracted.

Outages also pose problems for transactions that are distributed across nodes. See "Configuring for Distributed Network Transactions" for information on ensuring transaction integrity for distributed transactions.

Configuring FUP RELOAD Activity

`FUP RELOAD` commands are used to optimize database storage and access. They also generate a large amount of audit compared with typical activity. This can cause Extract to fall significantly behind the current location in the audit trails, sometimes requiring audit tape dumps to be restored. This process requires operator intervention.

You can often avoid tape restores by scheduling `FUP RELOADs` more effectively. Schedule reloads less frequently, or over several periods rather than all at once (For instance, reload 20% of the database each night for five nights, instead of reloading 100% of the database in a single evening.)

Data Compression

You can optionally configure Oracle GoldenGate to compress data before sending it over TCP/IP. The Collector automatically decompresses it on the target system. To compress records over TCP/IP, include the `COMPRESS` and `COMPRESSTHRESHOLD` options in the `RMTHOST` parameter statement.

- `COMPRESS` specifies that outgoing block of extracted changes are compressed, resulting in a typical 4 to1 ratio or better.

- COMPRESSTHRESHOLD sets the minimum byte size for which compression will occur. The default is 1000 bytes.

For TMF-audited Enscribe files, set the NonStop AUDITCOMPRESS file attribute when creating the file. For non-TMF files, specify the COMPRESSUPDATES argument in the Logger configuration.

Compressed Enscribe Records

Whether TMF or non-TMF, Enscribe compression transfers the following data (rather than sending all field values).

- Each fragment of the record that changed
- The key fragment of the record
- Four additional bytes per fragment indicating fragment position and length

The format of a compressed Enscribe record is as follows:

Field	Description
field offset	The offset within the original record of the changed value (2 bytes)
field length	The length of *field value* (2 bytes)
field value	The data, including null or varchar length indicators

The first field in a compressed Enscribe record is the primary or system key.

Compressed SQL Records

By default, SQL updates are compressed in the audit trails. This means each SQL update record includes the following data.

- Each column that was SET in the SQL UPDATE statement
- Each key column in each row updated
- Four additional bytes per column indicating column number and length

The format of a compressed SQL record is as follows:

Argument	Description
field index	The ordinal index of the SQL column within the source tables (2 bytes)
field length	The length of *field value* (2 bytes)
field value	The data, including null or varchar length indicators

DCOMPRESS File Attribute Not Supported

Turn off the NonStop DCOMPRESS file attribute for both SQL tables and Enscribe files extracted using TMF audit trails. When DCOMPRESS is on, compression occurs within each data block,

which prevents the resolution of entire record values. Extract is permitted, but unpredictable results can occur.

AUDITCOMPRESS File Attribute Considerations

When update operations occur on a file or table with audit compression on, only changed columns or fields and those that are part of the primary key are recorded. This means the full update images are not immediately available to Extract. Instead, a compressed image is retrieved and written.

This is acceptable for replication because only changes are required. However, problems can occur in the following circumstances:

- A selection clause includes columns that are not part of the source file's primary key.
- Columns are mapped, and the primary key of the target is different than that of the source.
- User exits or custom applications do not anticipate compressed records, which are more complex to process.

Extract provides an option to retrieve full record images from the original database. However, retrieving each update can slow processing considerably. The options you use, and whether you use audit compression, is based on your application's requirements.

The NonStop `AUDITCOMPRESS` attribute is controlled at the file and table level using FUP and SQLCI.

Configuring for Distributed Network Transactions

In a multi-node environment a single transaction may include changes to files on more than one node. For example, a customer's order may require updates to the customer file on `\A`, the customer account file on `\B`, and the order file on `\C`. Updates like these, as well as updates to tables that are partitioned across nodes, are referred to as distributed network transactions.

To help ensure the completeness of the transaction when one node experiences an outage, you should configure components that coordinate the updates for distributed network transactions. This avoids part of a transaction being committed while the changes going to a disabled node are lost.

The following processes play a part in this coordination. The required configuration setup is explained for each component.

- Manager

  When using a Coordinator, `PURGEOLDEXTRACTS` should be defined for the Manager rather than Replicat or Extract. This allows consideration of Coordinator checkpoints to ensure trail files are not purged before Coordinator has completed processing them. See "Recommendations for Managing Trail Purges" for more information.

  Also the Manager on the node where the Coordinator resides may optionally be configured to `AUTOSTART` the Coordinator process.

- Extract

  There are no configuration changes needed for Extract, but if it has the `PURGEOLDEXTRACTS` parameter, this should be moved to the Manager.

- Replicat

  The `COORDINATOR` parameter is added to the Replicat parameter file to define the name of the process that is coordinating its distributed transactions. When the Replicat encounters

a distributed transaction, it communicates with this Coordinator to determine when it can process that transaction.

If the Replicat has the `PURGEOLDEXTRACTS` parameter, it should be moved to the Manager to allow consideration of the Coordinator's checkpoints.

- Reader

  `READER` parameters are included in the `COORDINATOR` parameter file. These are used to configure Reader processes when the Coordinator is started.

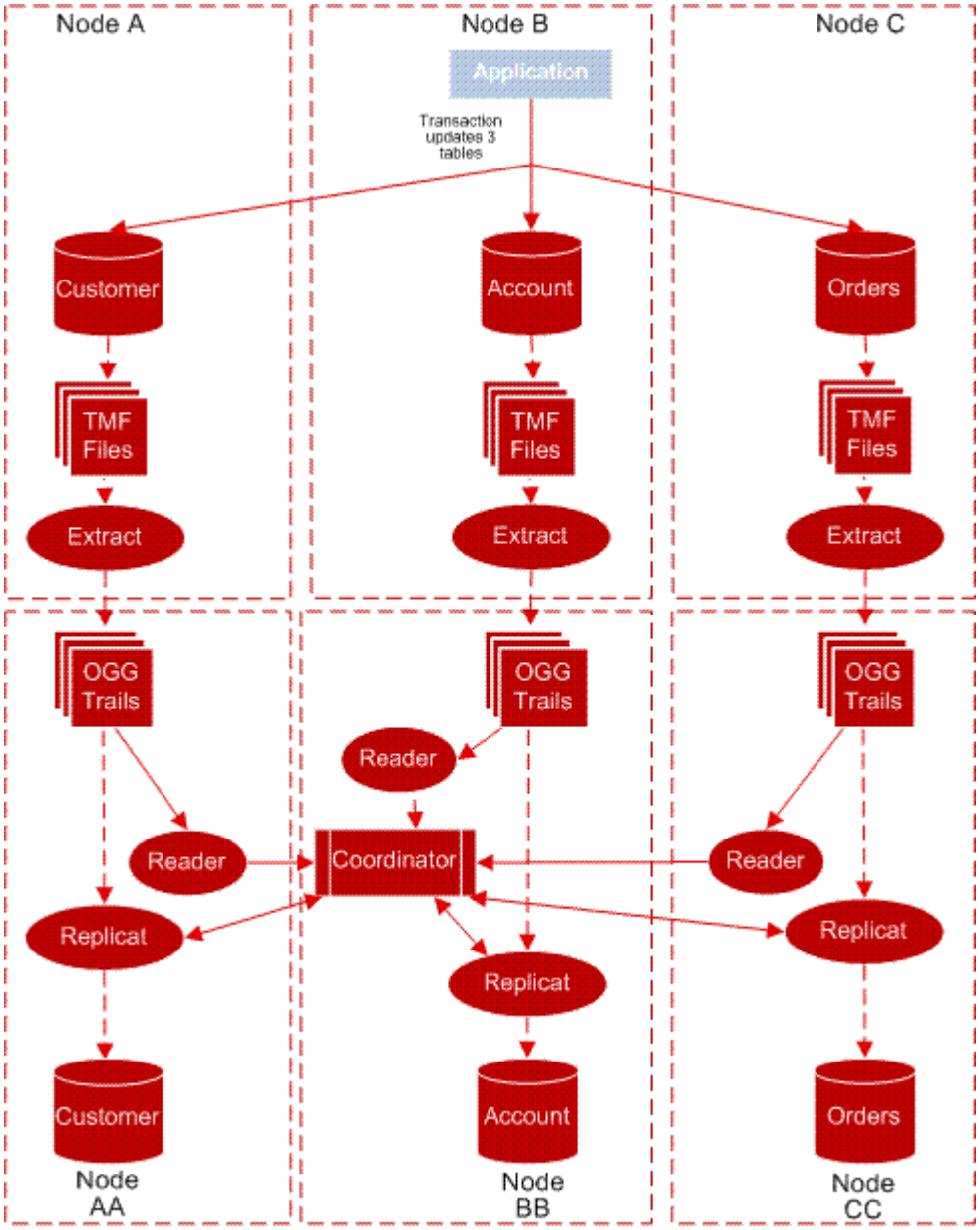
  The Reader scans the local Oracle GoldenGate trail for distributed transactions. When one is found, the Reader gathers local transaction information and sends it to the Coordinator process.

  > **Note**
  >
  > If Readers will not be configured because distributed network transactions do not need to be replicated, the Extract parameter `EXCLUDEGGSTRANSRECS` can be used. This will suppress the creation of trail records that track distributed network transactions.

- Coordinator

  A Coordinator process must be added on one of the nodes in the system. This is added using the GGSCI `ADD COORDINATOR` command. The parameter file for it includes `READER` parameters to establish the Reader process for each node and Oracle GoldenGate trail.

Figure 2-1 Process flow for distributed network transaction support



Example 2-1 Sample Coordinator Parameter File

```
COORDINATOR COORD1
FASTREADS
READER EXTTRAIL \NY.$DATA5.GGSDAT.AA, PROCESS $GGRD1, CPU 1, PRI 180
READER EXTTRAIL \LA.$DATA01.GGSDAT.BB, PROCESS $GGRD2
READER EXTTRAIL \FL.$DATA2.GGSDAT.CC, CPU 1, PRI 170
```

Coordinator receives information from the Readers, tallies the number of changes that have been received, and stores checkpoints. Coordinator uses this information to respond to queries from the Replicats on each of the nodes asking if the transaction is complete. When all of the operations for the transaction have verified their arrival, Coordinator releases the transaction to the Replicats for processing.

The following diagram shows an example of coordination processes for a distributed network transaction that spans three nodes, with each node replicated to a backup node.

Re-configuring TMF

When facts about the audit trails change, the checkpoints recorded by Extract can be invalidated, and TMF must be re-configured.

Before re-configuring TMF:

1. Use the GGSCI `INFO EXTRACT *` command to ensure that all Extract groups have processed through the end of the last audit file.
2. Use the GGSCI `DELETE ATCONFIG *` command to delete the current audit management parameters.
3. Delete all Extract groups.

After TMF is reconfigured:

1. Manually re-add all of the Extract groups.

Using TMFCOM, dynamically add and delete the volumes on which audit files are located. Deleting an `ACTIVE` or a `RESTORE` volume can have adverse effects. Before deleting a volume, make sure all groups have processed outstanding audit on that volume, or copy all files on that volume to the alternative location. After a volume is deleted, the Extract process and Manager will not be able to find the associated audit. You can add an `ACTIVE` or `RESTORE` volume with no impact on Extract operations.

Configuring Non-TMF-Enabled Processing

To capture data from non-TMF applications, you must bind GGSLIB to the user application. GGSLIB will intercept certain NonStop commands in the application's place, while Logger will write data to a log trail. This causes the following planning issues:

Maintaining Data Integrity

The following issues can cause GGSLIB and Logger to miss records and/or compromise data integrity:

- Log processes are stopped by an operator while the application is updating a database. Several safeguards are built in to deal with this potential problem.
- If a log process is stopped from TACL by process number, which can happen accidentally, the backup process takes over with no loss of data.
- If a log process is stopped from TACL by name, this is assumed to be a mistake (because the proper method is the GGSCI `STOP LOGGER` command). Manager immediately restarts log processes stopped this way, although records can be lost if this occurs while there is activity in the system.
- Double CPU failure occurs, taking down both the primary and backup log process CPUs. When this happens, other data integrity issues will surface on NonStop as a whole, such as loss of file buffers.
- Application programs are not bound with GGSLIB. This can happen when a program is omitted from the initial bind list. This can also happen when migrating new object code into production, then forgetting to perform the GGSLIB bind. To avoid this problem, include GGSLIB binding into version control procedures and check any programs that generate warnings (See "Authentication for Bound Programs" for more detail.)

- An application process is killed from TACL. This can mean that reads from or writes to the database could be lost in transit to the log process, depending on the timing of the STOP command. This is not a problem when issuing FREEZE and STOP commands to Pathway servers.
- Extract or Replicat processes fall far behind Logger. Eventually, log trails are recycled by Manager, regardless of whether they are required by Extract or Replicat. EMS warnings can be generated to alert operators to this condition. This most likely happens when a network or target system is down for an extended period.

Supported File Types and Operations

GGSLIB and Logger behave according to the following rules regarding file operations.

- The following file types are supported: Key-sequenced, entry-sequenced, queue-files, syskey-files, relative and unstructured file operations. However, updates to edit files and the spooler cannot be extracted. Unstructured files must be extracted explicitly (using the GETUNSTRUCTURED parameter in the Logger parameter file).
- Bulk I/O operations, i.e. operations that use SETMODE, are supported. The current list of SETMODEs includes:
  - 1 - Set file security
  - 2 - Set file owner
  - 3 - Set write verification
  - 57 - Set serial writes
  - 90 - Set buffered
  - 92 - Set maxextents
  - 93 - Set unstructured buffer length
  - 94 - Set auditcompress
  - 97 - Set licensed
  - 123 - Set generic lock key length
  - 138 - Set/Reset corrupt
  - 153 - Set variable length audit compression

  FUP DUP, FUP LOAD and SELECT n AREA in COBOL programs are also included.
- To extract bulk I/O operations, specify the GETBULKIO option in the Logger parameter file. FUP COPY is supported by default. Use GETFILEOPS in Extract and Replicat to propagate these operations to the target database.
- FILE ALTER, CREATE, DUP, PURGE, PURGEDATA, and RENAME operations (to disk files) are supported.
- The following CONTROL operations are supported:
  - 2 - set end-of-line
  - 20 - PURGEDATA
  - 21 - Allocate/Deallocate extents
- Use GETFILEOPS in Extract and Replicat to propagate the operations listed above to the target database.

- Undocumented, privileged function calls used by `FUP DUP` and `FUP LOAD` to change file labels are supported (This requires `PRIVLIB` to be licensed and included as Replicat's user library.) These functions are required to fully implement `FUP DUP` and `FUP LOAD` of key-sequenced files.

Authentication for Bound Programs

An exit can be activated within NonStop Safeguard to access the Oracle GoldenGate module SFGEXIT. This program runs as an independent process to monitor non-audited file opens for update access. (Opens for audited files or SQL tables and read-only opens are ignored.) When a non-audited open is found, SFGEXIT determines if the opening program has the Oracle GoldenGate intercept library bound to it. If it does not, the following warning is issued to EMS to alert the user that updates may occur without replication.

```
GoldenGate Library is not bound to $vol.subvol.program_name and it may
update $vol.subvol.application filename
```

System Utilities That Update Databases

Standard NonStop utilities, notably FUP and TACL, perform file operations such as `CREATE`, `COPY`, `PURGE`, `PURGEDATA`, `DUP`, `LOAD`, and `RENAME`. You can monitor these activities by binding GGSLIB to these utilities just as you would to an application program.

Private Memory and Stack Space

GGSLIB routines minimize stack space requirements. By doing so, programs are ensured there will be enough stack room for typical activities.

For its own working space, GGSLIB allocates a small private memory segment to handle in-transit I/O buffers and keep its own state variables.

Impact on Existing Application Performance

GGSLIB and Logger add a small amount of overhead to existing application activities. Messages to log processes are sent asynchronously (`NOWAIT`) to avoid making the application wait for logging to occur. In addition, log processes write all data sequentially into buffered files for the best possible performance.

Configuring Oracle Goldengate Global Operations

User can configure Oracle GoldenGate global operations.

GLOBALS Parameter File

Oracle Goldengate provides the `GLOBALS` parameter file to standardize Oracle Goldengate configuration. Typically, you set global parameters when you install Oracle Goldengate. Once set, you rarely need to change them. Some of the operations you can standardize are:

- The initial allocation for wildcard entries
- The time out value when GGSCI communicates with Oracle GoldenGate components
- NonStop nodes in the network
- The refresh interval

- TACL DEFINEs for GGS_AUDCFG and GGS_PREFIX when not using the default

To support versatility, some of the parameters set in GLOBALS can be temporarily overridden by other Oracle GoldenGate programs.

See Oracle GoldenGate Parameters for more information about global parameters.

Changing the Default Location of AUDCFG

Run BUILDMAC or NLDLIB to change the default location where an instance of BASELIB, GGSLIB, GGSSRL, or GGSDLL will look for the AUDCFG segment. When it builds the new library, the macro prompts to ask if you want to change the AUDCFG location. If the answer is yes, you will be prompted for the new default $VOL.SUBVOL location.

If you want multiple Oracle GoldenGate environments to each have a different location for the AUDCFG segment, each environment will need a unique copy of GGSLIB or BASELIB linked with the location specific to that environment.

If the library specifies a different location for the AUDCFG than the DEFINES included in the GLOBALS parameters, the GLOBALS DEFINES will override the library.

Configuring Replication

Replicat provides a high degree of flexibility when processing data between files; however, there can be logical restrictions involved for which you must plan. This section details different scenarios that require additional planning, including:

Replicating SQL Tables with System Keys

Entry-sequenced SQL tables with non-unique keys are sometimes difficult to replicate accurately. This is because their keys are a SYSKEY value generated by the system. Replicat has no control over the SYSKEY value when replicating an insert operation into the target table; therefore subsequent update and delete records cannot be replicated exactly. Even though the SYSKEY value of the original record is known, the replicated record has a different SYSKEY value, requiring you to create a workaround so your keys resolve properly.

There are two methods for working with this issue. You can specify a view that contains all columns from the base table excluding the SYSKEY. Use the view as the target in the replication MAP, along with a KEYCOLS specification to define a different method for accessing the table for delete and update operations. This requires each target row to have some type of unique identifier, such as a unique index.

Another method is to add a column called GGS_SYSKEY to your target table, then map the source SYSKEY value to the GGS_SYSKEY column. Specify GGS_SYSKEY in the KEYCOL option of the map argument and use the FORCEUSESYSKEY parameter.

Replicating Primary Key Updates

Although Nonstop Enscribe and SQL/MP do not allow changes to primary keys, operations for primary key updates may be received from Oracle GoldenGate systems running for other databases. To maintain compatibility, Oracle GoldenGate for NonStop processes these primary key update operations by deleting the record and then inserting it with the same data, but a new primary key.

Primary key updates for Enscribe entry-sequenced and queue files are not supported and will generate an error.

The default is to process primary key updates, but a parameter is available to turn this off and discard the record. Contact *Oracle GoldenGate Technical Support* to use this parameter.

Missing Row Errors

Because values are needed for the columns that were not changed, an error will occur if the record cannot be fetched from the target database.

If `HANDLECOLLISIONS` is turned on and the fetch fails, there is an attempt to insert the missing record. Otherwise if `REPERROR` responses have been defined for a missing row, the rules specified by the `REPERROR` will be applied.

Non-Audited Target

An error message is returned if an unaudited Enscribe record is deleted and then the insert of the new primary key record fails. Because it is not possible to back out the records processed since the last checkpoint, the system will advance the checkpoint to the record that is in error. User intervention will be required to correct the target record and restart the Replicat.

- For a file system error, correct the cause of the problem and insert the record from the discard file. Then skip over the primary key update record by advancing the checkpoint RBA to the next record.
- If the insert generates a duplicate error, try to determine if the discarded record is more correct than the target record. If it is, delete the record in the file and replace it with the discarded record. Then skip over the primary key update record by advancing the checkpoint RBA to the next record.

Compressed Updates to Enscribe Targets

`TARGETDEF` using `DICTIONARY` or `SOURCEDEFS` is required when:

- Compressed updates are being replicated to an Enscribe target database.
- For sending Enscribe data to OpenSys, see Oracle GoldenGate Self Describing Trail Files

Files and Tables Other Than Key-Sequenced

You can replicate files and tables that are not key-sequenced, but there will be conditions that apply.

For relative files, Oracle GoldenGate forces the relative key of the target file to be the same as the source, so target records can be found for updates and deletes. The condition is that you can only replicate from a single source to a single target.

You have more flexibility if the relative file or table has a unique index. Then the columns in that index can be specified with `KEYCOLS` to identify a path for update and delete statements. However, any application that stores system keys as foreign keys in other tables will have unreliable results.

For entry-sequenced files or tables, selective replication (that is, where selection criteria are applied) is only feasible for inserts. This is due to the difficulty identifying the correct target record for updates. Selective replication from one source to one target is feasible for relative files and tables.

Entry-sequenced files can be replicated in the same order when the source database is TMF audited because the TMF data is in the correct order. If the source database is non-TMF, and GGSLIB is used to extract the data, records may be written to the target file in a different order

than they appear in the source. This has a corresponding effect when updates to entry-sequenced records are processed: the record address of the source may be different from that in the target, resulting in a missing or incorrect update.

To get around this, when replicating a non-TMF entry-sequenced file from one source to one target, you can use the parameter and option `ENTRYSEQUPDATES EXACTKEY`. This requires the target file to be opened with `PROTECTED` or `EXCLUSIVE` access so other processes (including other Replicats) can not update the file. See Oracle GoldenGate Parameters for more information on how to use this parameter.

See "Bi-Directional Replication" for information on an environment not limited to single source updating a single target.

Load Balancing and Performance Issues

Replicat often proves to be a bottleneck when initially configured, especially for hot site applications that replicate the entire database. This bottleneck is because Replicat often mimics the original application's processing. In general, this may mean many more random, unbuffered I/Os. In contrast, Extract and Logger perform serial, buffered I/Os, usually in large blocks.

To solve this problem, configure multiple Replicat processes, each of which replicates a portion of the overall data.

One way to do this is assign different files or tables to different Replicat processes. This is conceptually simple. For example, if an application consists of data in four tables, `TAB1`, `TAB2`, `TAB3`, and `TAB4`, let Replicat process #1 replicate `TAB1` and `TAB2`, while Replicat process #2 replicates `TAB3` and `TAB4`.

A more complex option is to split the same file or table among multiple Replicat processes. This might be necessary, for example, when one million inserts and updates per day might occur against `FILE1`, while in the rest of the system only 100,000 inserts and updates occur. In this case, the optimal configuration may be two Replicat processes for `FILE1`. This is accomplished in two steps:

1. Let Extract split the data into two trails. Each trail contains half the data for `FILE1`. To split the data, use the `WHERE`, `RANGE`, or `FILTER` clause of the Extract file parameter.
2. Assign a Replicat process to each of the resulting trails.

Example 2-2 Splitting to Two Trails

```
EXTRACT DEMO
EXTTRAIL \NY.$DATA1.GGSDAT.E1
TABLE $DATA.MASTER.ACCOUNT, WHERE (ACCOUNT < 500000);
EXTTRAIL \NY.$DATA3.GGSDAT.E2
TABLE $DATA.MASTER.ACCOUNT, WHERE (ACCOUNT >= 500000);
```

A Replicat group is then dedicated to process each of the trails above.

Splitting up tables among different Extract processes may temporarily upset original transaction integrity boundaries, because two or more processes may be replicating a single transaction.

The following Extract parameter file splits `$DATA.MASTER.ACCOUNT` into two trails.

Potential Problems with Audit Compressed Files

When replicating records selected with `WHERE` criteria from a source file with audit compression, update records can be missed (deletes and inserts will always be extracted). You can guarantee that all updates are processed by omitting fields that are not part of the primary key from your `WHERE` clauses. Primary key fields are always present in compressed update records.

When mapping selected columns with `COLMAP`, audit compression also causes potential conflicts. If the key of the target file includes a field not contained in the key of the source, target updates can fail. Updates require the presence of the entire key to guarantee success.

The easiest method for avoiding these conflicts is to turn off audit compression for source tables and files. This may or may not be feasible depending on the characteristics of your transaction load.

Conflicts with Updating the Target

If both Oracle GoldenGate and another application are allowed to update a target, conflicts can arise unless you establish rules to avoid them. For example, application #1 might update a record in the source database that application #2 has deleted from the target database. In such cases, it is impossible for Oracle GoldenGate to apply the source update at the target because the record to update no longer exists.

As a general rule, Replicat should have control over ranges of data that other applications cannot update. However, if conflicts are tolerable, Oracle GoldenGate provides features that allow operations to continue uninterrupted when errors occur:

- Use the `REPERROR` (*error*, `IGNORE`) parameter entries to ignore errors that otherwise cause transactions to abort.
- Use `OVERRIDEDUPS` and `INSERTMISSINGUPDATES` to ensure all updates are inserted.
- Review the Replicat discard file for operations that failed, and determine corrective measures.

Many-to-One Replication

When replicating many files to one file (collecting), applications should ensure that each source file manages a specific range of keys. If different source files can update the same key value, there can be conflicts at the target. For example, if two source tables receive an insert with the same key, both operations cannot be applied at the target because a duplicate error will result (Guardian error `10`, SQL error `-8227`).

Oracle GoldenGate provides several alternatives for dealing with this problem. One is the `HANDLECOLLISIONS` parameter that directs Replicat to insert the latest version of the record, even if the key exists. `HANDLECOLLISIONS` ignores missing update and delete conditions. Another option is to restrict the range of values replicated from each source with `WHERE` criteria. Most often the best alternative is to avoid the possibility of such conflicts as part of the application's processing rules.

Bi-Directional Replication

Sometimes, you may want to have two or more files replicating data to each other. In such cases, have each file manage a unique range of keys directly, as in the many-to-one case above. The difference here is that each file will hold data it manages, along with data replicated

from the other file. In this way, each file can act as a backup for the other. The application should ensure that replicated data is read-only in such cases.

Because both files must be replicated, each replicated change will itself be extracted and replicated back to its source, which will cause errors. There are two methods for avoiding this condition:

- Restrict the ranges of key values that are extracted and replicated using `WHERE` criteria.
- Use the `IGNOREREPLICATE` parameter in Extract processing. This parameter causes Extract to discard any operations that were applied by Replicat processes.

> **Note**
>
> `PURGEDATA` is a DDL statement that is automatically committed and not linked to any transaction, so `GETPURGEDATAS` is not supported for audited files in a bidirectional configuration. In this case loop detection is not effective and `IGNOREPURGEDATAS` must be included in the Extract parameters.

Replicating Data to Non-TMF Enabled Databases

You can stimulate overall system performance by implementing buffering on your non-TMF Enscribe databases. To do so, turn on file buffering for target database files with the `FUP ALTER` *`filename,`* `BUFFERED` command. This imposes no real risk because the data is mirrored at the source system and can be recovered from there.

Use the `NOAUDITREPS` Replicat parameter to avoid unnecessary event messages regarding non-audited target files.

Replicating New SQL Columns

To replicate new SQL columns that were created since the current Extract and Replicat processes were started, include `REPNEWCOLUMNS` in the Replicat parameter file. `REPNEWCOLUMNS` replicates the `SQL ALTER TABLE ADD COLUMN` statements to create the new columns in the target.

Alternatively, you can specify `GETNEWCOLUMNS` to update table definitions when a column change is detected on a source table. `GETNEWCOLUMNS` ensures that data in columns created after Replicat starts up (using `ALTER TABLE ADD COLUMN` on the source system) are accounted for.

Configuring for Maximum Throughput

You can maximize throughput by modifying Extract, Replicat, or both. This section details strategies for implementing Oracle GoldenGate parameters to achieve data management that suits your needs.

Extraction

Techniques for maximizing throughput on Extract depends on whether the source system produces TMF trails or non-TMF logs.

TMF Extraction

In most cases, only a single instance of Extract is required to extract and transmit data to the target system. A single Extract is advantageous because TMF audit trails are only read once.

In rare cases, extracting high volumes of `SQL UPDATE` statements requires multiple instances of Extract.

Non-TMF Data Extraction

Non-TMF logging is linearly scalable by adding more Logger processes to the configuration. Because there is no penalty for adding Logger processes to the configuration, Oracle GoldenGate recommends allocating plenty of slack for high volume activity. In most cases, two or three Logger processes is more than enough to achieve the desired throughput.

Replication

To achieve required throughput, more Replicat processes may be required. This is because Replicat's I/O activity tends to be random access, as opposed to Logger and Extract I/O, which is serial, blocked and buffered.

You can add Replicat processes to achieve near linear performance gains. However, to ensure good performance, no more than three Replicat processes should read each Oracle GoldenGate trail. Otherwise, excessively redundant reads result, sometimes causing contention issues on the trail's disk.

Latency Issues

Latency often refers to the difference in time between when an update occurs on the source database and when that same update is replicated on the target database. In this respect, latency measures the amount of time "behind" that the target system is from the source system, and can be important when determining the target database's accuracy. Database latency is especially important in certain bi-directional scenarios when two systems might update the same record in different databases at virtually the same time.

Another measure of latency is the lag between an update on the source and the time at which that update has been stored on the target system for later replication. This measure of latency represents the potential for the amount of data lost in a disaster. Once data has been transmitted to the target, it will be replicated eventually and is not exposed to the risk of disaster.

Capacity Planning

Through testing, Oracle GoldenGate has compiled some capacity planning guidelines, presented in the following sections. Consider these observations as guidelines; actual performance depends on many of the factors previously discussed including network topology, operating systems, etc.

TMF Data Extraction

Oracle GoldenGate output figures are far less than the audit generated, because extracted data does not include alternate keys, SQL indexes, `FUP RELOAD` information and assorted audit records.

Non-TMF Data Extraction

Non-TMF extracts are linearly scalable. Therefore, the potential extraction rate of data is close to the system limits for existing application activity.

Data Transfer into Oracle GoldenGate Trails

The potential for data transfer is around 75-80% of the communication channel's actual potential. When this limit is reached, you can split data into multiple trails to achieve greater throughput with parallelism.

Replicat Throughput

The potential throughput of Replicat is greater than that of the database I/O performed on the source system. Replicat performs essentially the same I/Os on the target system as were performed on the source system, excluding reads. In addition, Replicat uses transaction grouping features as mentioned earlier to improve TMF-related performance

Changing Default Component Names

GGSCI provides default names for processes, parameter files, and report files. You may want to change these defaults to make them more descriptive. For example, you may want to denote the parameter files and reports associated with a particular Extract or Replicat group (when have multiple Extracts and Replicats).

To change default component names:

1. Launch GGSCI.
2. Specify the define `=GGS_PREFIX` using the following syntax.

   ```
   GGSCI> ADD DEFINE =GGS_PREFIX, CLASS MAP, FILE $prefix
   ```

   Where:

   prefix consists of two letters.

   Consider the example:

   ```
   GGSCI> ADD DEFINE =GGS_PREFIX, CLASS MAP, FILE $EF
   ```

   This example changes the following default components.

   - The Manager process name changes from `$GGMGR` to `$EFMGR`.
   - Logger process names become `$EFLnn` instead of `$GGLnn`.
   - Parameter files are stored in the `EFSPARM` subvolume rather than `GGSPARM`.
   - Report files are stored in the `EFSRPT` subvolume rather than `GGSRPT`.
   - Extract processes are called `$EFEnn` rather than `$GGEnn`.
   - Replicat processes are called `$EFRnn` rather than `$GGRnn`.
   - Syncfile processes are called `$EFSnn` rather than `$GGSnn`.
   - Coordinator processes are called `$EFCnn` rather than `$GGCnn`.
3. One way to tell GGSCI and application programs bound with GGSLIB where to establish and retrieve configuration information is to use the `=GGS_AUDCFG` define. Alternatively the

location can be specified when running BUILDMAC or NLDLIB. When this location is not provided with one of these methods, the default is $SYSTEM.GGS.AUDCFG.

```
GGSCI> ADD DEFINE =GGS_AUDCFG, CLASS MAP, FILE config_file
```

Where:

config_file is a file name, and the file portion of the file name is no longer than six characters.

Using Wildcards

You can use wildcard arguments to express volumes, subvolumes, files and tables. However, you can't use wildcard arguments to express views. Oracle GoldenGate allows wildcards to be expressed as a question mark (?) or an asterisk (*). An asterisk matches any number of characters, whereas a question mark matches only a single character.

The wildcard expression in the following example refers to any file set in the specified volume and subvolume:

```
FILE $DATA1.MYSUB.*;
```

In this next example, the wildcard expression refers to any volume $DATAn, where n represents the fifth character in the volume name, and any file in the specified subvolume:

```
FILE $DATA?.MYSUB.*;
```

By default, Oracle GoldenGate initially allocates 100 wildcard entries. You can change this initial allocation using the MAXWILDCARDENTRIES parameter in the GLOBALS, Extract, and Replicat parameter files. Once this initial MAXWILDCARDENTRIES allocation is exhausted, the program will allocate an additional 100 entries each time it needs more.

When you specify MAXWILDCARDENTRIES in the GLOBALS parameter file, that specification becomes the default. You can override that default using the MAXWILDCARDENTRIES parameter in the Extract or Replicat parameter files. Ensure that a View exists before the Extract is started.

Most parameters that specify file names or table names can use wildcard expressions. Exceptions are documented in the parameter's description.

Support for DDL and DDL2

Oracle GoldenGate for HP NonStop supports Enscribe Data Definition Language dictionary builds for DDL or DDL2. The versions that are supported include the following:

- C20 and C30 operating system, DDL Version 5
- D00, D10 and D20 operating system, DDL Version 6
- D30 and later, DDL Version 7
- H01 and later, DDL Version 8 and DDL2 Version 9

An error will be generated if a valid version is not identified.

No parameters are required for DDL2 support. Definitions of either size are supported for column mapping. User exits support record manipulation on the DDL2 large record formats by using a new set of function calls. See the user exit function calls in the Oracle GoldenGate Parameters for details.

Specifying Internet Protocol Addresses

Manager and Extract can be restricted to a specific IP address by using the `IPINTERFACE` stand-alone parameter or the *`@ip_address`* option of `TCPIPPROCESSNAME`.

- This example using the `IPINTERFACE` stand-alone parameter sets the IP address to `2001:db8:2010:5040:4fff:ffff:ffff:28`.

  ```
  IPINTERFACE 2001:db8:2010:5040:4fff:ffff:ffff:28
  ```

- This example using the *`@ip_address`* option of `TCPIPPROCESSNAME` parameter sets the process name to `$ZTC4` and its IP address to `2001:db8:2010:5040:4fff:ffff:ffff:28`.

  ```
  TCPIPPROCESSNAME $ZTC4@2001:db8:2010:5040:4fff:ffff:ffff:28
  ```

Targets can be restricted using options of the `RMTHOST` parameter.

- This example using the `IPINTERFACE` option sets the IP address of the host to `2001:db8:2010:5040:4fff:ffff:ffff:28`

  ```
  RMTHOST host01, MGRPORT 12345,
  IPINTERFACE 2001:db8:2010:5040:4fff:ffff:ffff:28
  ```

- This example using the *`@ip_address`* option of `TCPIPPROCESSNAME` sets process name to `$ZTC1` and IP address to `2001:db8:2010:5040:4fff:ffff:ffff:28`.

  ```
  RMTHOST host01, MGRPORT 12345,
  TCPIPPROCESSNAME $ztc1@2001:db8:2010:5040:4fff:ffff:ffff:28
  ```

See the Oracle GoldenGate Parameters for more details on how to use these parameters.

Oracle GoldenGate for HP NonStop supports Internet Protocol versions 4 and 6 (IPv4 and IPv6.) If an IP address is specified for Manager or Extract, the matching version must be configured for that host or an error is generated. If a name is specified for the host and both IPv4 and IPv6 are configured for that host, the default is to use IPv6. The parameter `USEIPV4ONLY` forces Extract to use IPv4.

3
Configuring Initial Data Synchronization

Before running the Oracle GoldenGate for Nonstop for the first time, synchronize the source and target databases. Here we address initial data synchronization when the source is TMF-Enabled.
This topic includes the following sections:

Initial Data Synchronization

You can use Oracle GoldenGate to load data in any of the following ways.

- Using database utilities. The utility program performs the initial load. Examples include loading the target using FUP or SQL and using back up and restore.
- Loading from a file to a database utility. Extract writes records to an extract file in external ASCII format. The files are used as input to a bulk load utility that writes to target. Replicat creates the run and control files.
- Loading from a file to Replicat. Extract writes records to an extract file and Replicat applies them to the target tables.
- Using an Oracle GoldenGate direct load. Extract communicates with Replicat directly without using a Collector process or files. Replicat applies the data to the target.
- Using direct bulk load. Extract writes records in external ASCII format and delivers them directly to Replicat, which delivers them to the Oracle SQL*Loader bulk load utility. This is the fastest method of loading Oracle data with Oracle GoldenGate.

The method you choose depends, in part, on the database types of your source and target. For example, if you are replicating from a NonStop source to an Oracle target, your choices include direct bulk load, which interacts with SQL*Loader.

Regardless of the method chosen, the initial data synchronization should be run after initiating change capture and before configuring delivery. The steps are:

1. Configure and run Extract to capture all active database changes to an Oracle GoldenGate trail.
2. Perform initial load.
3. Configure and run Replicat.

Example Steps for Initial Data Load

This example shows how you might follow these three steps to configure change capture and delivery and perform an initial data load.

Configure and Run Extract

You can configure this Extract process to read change records from a TMF audit trail, log trail, or flat file, and process them to an Oracle GoldenGate trail. This step is necessary before doing the initial load because it extracts and delivers ongoing changes to the source database, preserving data integrity for your business operations.

Instructions for configuring and running Extract can be found in "Planning the Configuration".

Perform Initial Load Using the File to Replicat Method

You can perform your initial load using any of the methods; however this example addresses how to use Oracle GoldenGate to do the initial load by queuing data to an Oracle GoldenGate file that will be picked up by Replicat.

1. Create one Extract parameter file to read directly from each source database.
2. Use the NonStop text editor to set up an Extract parameter file to include the following information.
   - The `SOURCEISFILE` parameter to indicate that data should be retrieved directly from the table.
   - The format for the target file, usually `FORMATASCII` (for example, `FORMATASCII`, `SQLLOADER` or `FORMATASCII`, `BCP`).
   - If you are transmitting data to a system other than NonStop, or to a NonStop system over TCP/IP, include the name of the remote TCP/IP host and port number of the remote Collector.
   - The name of the local output file (`EXTFILE`) or the remote file (`RMTFILE`) to which the Extract program writes extract information. If you need to write to a series of trails, then add the `MAXFILES 2` to the remote trail's parameter file. `MAXFILES` will append a six-digit trail sequence to the remote trail's file name.
   - The name of the file or table to extract (`FILE` or `TABLE` parameter).
   - Other optional parameters, including clauses for selecting records, column mapping, or data conversion.
3. Configure Replicat as a batch task, specifying the `SPECIALRUN` and `BEGIN` and `END` parameters.
4. Start the initial data load:

```
TACL> RUN EXTRACT /IN parameter_file/
TACL> RUN REPLICAT /IN parameter_file/
```

Configure and Run Replicat

When your initial data load finishes writing to its trails, configure Replicat on the target system. This can be the same parameter file you use for ongoing Replicat work, however, you will need to add `HANDLECOLLISIONS` and `END` parameters, run the batch, then remove those parameters before beginning ongoing change extract.

1. Configure Replicat on the target system, including `HANDLECOLLISIONS` and `END` in the parameter file. The `END` parameter is the time recorded for the completion of the Extract process.
2. Start Replicat with the `START REPLICAT` command.
3. When Replicat stops, remove the `HANDLECOLLISIONS` and `END` parameters.
4. Start Replicat for incremental data synchronization.

Direct Load

Using direct load, you can extract data directly from the source tables and send it, in a large block, to Replicat. You may do this on any Operating System and database combination Oracle

GoldenGate supports (such as NonStop to NonStop, NonStop to OpenSys (Classic), and OpenSys (Classic) to NonStop). This is not supported to or from OpenSys MicroServices.

To run direct load:

1. Define an Extract group:

   ```
   GGSCI> ADD EXTRACT group_name, SOURCEISFILE
   ```

2. Define a Replicat group:

   ```
   GGSCI> ADD REPLICAT group_name, SPECIALRUN
   ```

   Replicat is automatically started by Manager, at Extract's request.

3. Create the parameter files.

   For the Extract parameter file:

   ```
   EXTRACT INITEXT
   RMTHOST targethost, MGRPORT 7809
   RMTTASK REPLICAT, GROUP INITREP
   TABLE $DATA.MASTER.ACCOUNT, AUTOTRUNCATE;
   TABLE $DATA.MASTER.PRODUCT, AUTOTRUNCATE;
   TABLE $DATA.MASTER.CUSTOMER AUTOTRUNCATE;
   ```

   - `AUTOTRUNCATE` sends a `PURGEDATA` command to Replicat before any data is processed. This ensures the target is clean and ready to receive data.

     > **Note**
     >
     > Use `AUTOTRUNCATE` with extreme caution, as it causes all existing data to be purged from the target file. Refer to *Reference for Oracle GoldenGate on HP NonStop Guardian* for more information.

   - `RMTHOST` establishes the remote TCP/IP host and port number.
   - `RMTTASK` instructs Manager on the target system to start Replicat with the specified `GROUP` name.
   - The `TABLE` parameters identify the source tables.
   - Specify `SOURCEISFILE` in the parameter if you want to include a `SOURCEISFILE` option:
     - `SELECTVIEW`: Selects data from a specified SQL view in the file parameter. Without `SELECTVIEW`, Extract selects data from the base table of the view, then maps the base table columns to the view columns (This also occurs when processing audit trails and a view is specified.)
     - `FASTUNLOAD`: Processes the file or table several times faster than the default method. Records are written out in random order, rather than primary key order. `FASTUNLOAD` has no effect when an SQL view is specified. The file parameter option `PARTITIONS` can restrict the data retrieved to a certain subset of the file or table
     - `FASTUNLOADSHARED`: Allows a shared open of the source file or table. This only required on files that are being updating by applications at the same time data is being extracted.

   For the Replicat parameter file:

   ```
   REPLICAT INITREP
   USERID GoldenUser, PASSWORD pass
   ```

```
SOURCEDEFS $DATA.DIRDEF.SRCDEF
MAP $DATA.MASTER.ACCOUNT, TARGET $DATA3.MASTER.ACCOUNT;
MAP $DATA.MASTER.PRODUCT, TARGET $DATA3.MASTER.PRODUCT;
MAP $DATA.MASTER.CUSTOMER, TARGET $DATA3.MASTER.CUSTOMER;
```

In the above example:

- `USERID` and `PASSWORD` are required to access the target database.
- `SOURCEDEFS` identifies the file containing the source data definitions.
- The `MAP` parameters map the source tables to the target tables, based on the data definitions in `SOURCEDEFS`.

4. Start Extract:

```
GGSCI> START EXTRACT INITEXT
```

Using Wildcards

Wildcards can be used for the `FILE` and `TABLE` statements in direct load parameter files, but not for views.

Refer back to the example of an Extract group added with the `SOURCEISFILE` parameter in "To run direct load:". If the `ACCOUNT`, `PRODUCT` and `CUSTOMER` files are the only files on `$DATA.MASTER`, the Extract parameters could be changed to use wildcards. This use of wildcards is shown in the following direct load parameter file:

```
EXTRACT INITEXT
RMTHOST targethost, MGRPORT 7809
RMTTASK REPLICAT, GROUP INITREP
TABLE $DATA.MASTER.*, AUTOTRUNCATE;
```

Direct Bulk Load

If you are loading to an Oracle target, you may choose to use direct bulk load. Direct bulk load is the fastest technique for capturing and delivering data to SQL*Loaders. Extract sends the data, in a large block, to Replicat. Manager dynamically starts Replicat, which communicates directly with SQL*Loader using an API.

> **Note**
>
> You can only use this direct bulk load from NonStop to Oracle Classic.

To run direct bulk load:

1. Define an Extract group:

```
GGSCI> ADD EXTRACT group_name, SOURCEISFILE
```

2. Define a Replicat group:

```
GGSCI> ADD REPLICAT group_name, SPECIALRUN
```

Replicat is automatically started by Manager, at Extract's request.

3. Create the Extract and Replicat parameter files.

Following are the examples of sample direct bulk load parameter files.

Sample Extract parameter file:

```
EXTRACT INITEXT
RMTHOST targethost, MGRPORT 7809
RMTTASK REPLICAT, GROUP INITREP
TABLE $DATA.MASTER.ACCOUNT;
TABLE $DATA.MASTER.PRODUCT;
TABLE $DATA.MASTER.CUSTOMER;
```

- RMTHOST establishes the remote TCP/IP host and port number.
- RMTTASK instructs Manager on the target system to start Replicat with the specified group name.
- The TABLE parameters identify the source tables.

Sample Replicat parameter file:

```
REPLICAT INITREP
USERID GoldenUser, PASSWORD pass
BULKLOAD
SOURCEDEFS /GGS/DIRDEF/SRCDEF
MAP $DATA.MASTER.ACCOUNT, TARGET master.account;
MAP $DATA.MASTER.PRODUCT, TARGET master.product;
MAP $DATA.MASTER.CUSTOMER, TARGET master.customer;
```

- USERID and PASSWORD are required to access the target database.
- BULKLOAD tells Replicat that SQL*Loader will load the target tables.
- SOURCEDEFS identifies the file containing the source data definitions.
- The MAP parameters map the source tables to the target tables, based on the data definitions in SOURCEDEFS.

4. Start Extract:

```
GGSCI> START EXTRACT group_name
```

Synchronizing Nonstop Databases Using Database Utilities Through TCP/IP

You can synchronize two NonStop tables or files on different systems over a TCP/IP connection using the trail to database utility method. Use the following steps:

1. Start the Collector on the target system.
2. Create a parameter file to perform initial file extraction over TCP/IP.

   Sample Extract parameter file:

   ```
   SOURCEISFILE, FASTUNLOAD
   FORMATLOAD
   RMTHOST 192.0.2.12, PORT 7829
   RMTFILE $D3.INIDAT.TRANSTAB, PURGE
   FILE \SRC.$D2.MYDB.TRANSTAB;
   ```

   For this example parameter file, named GGSPARM.TRANINI, you are identifying the Collector on the remote system.

   - SOURCEISFILE, FASTUNLOAD directs Extract to retrieve data directly from the blocks of the table.
   - FORMATLOAD directs Extract to format the data compatible with FUP or SQLCI LOAD.

- RMTHOST identifies the IP address and port of the Collector process. When NonStop is the receiver, a separate Collector is required for each simultaneously active session.
- RMTFILE identifies a flat file that holds the extracted data until the load is finished.
- FILE identifies the source file or table from which to extract the data.

3. Run Extract to extract the data into a flat file on the target system.
4. Use FUP or SQLCI to insert the data into the target system, similar to:

```
SQLCI> LOAD $D3.INIDAT.TRANSTAB, $D4.MYDB.TRANSTAB, RECIN 236, RECOUT 236;
```

The figures for RECIN and RECOUT are derived from Extract's recordings in $S.#TRAN, which includes the physical length of the records in the target. For Enscribe, this is the same as the record length returned by FUP INFO. For SQL, the size will vary and can be returned from RECORDSIZE column in the FILE table from the source table's catalog.

Controlling the IP Process for Replicat

Although you can configure multiple initial-load Extracts and Replicats, by default the Replicats will inherit the IP process of the Manager running on the target. This results in a single IP channel that does not spread the load across the processors.

To configure the Extract and Replicat pairs to use different channels, you can use static Replicats as shown in the next example.

1. Configure multiple Extracts to use the PORT parameter, rather than MGRPORT. Assign a different port to each.

```
EXTRACT extl1
RMTHOST 192.0.2.1, PORT 12345
RMTTASK REPLICAT, GROUP repl1
EXTRACT extl2
RMTHOST 192.0.2.2, PORT 12346
RMTTASK REPLICAT, GROUP repl2
```

2. Start static Replicats using the run-time parameter INITIALDATALOAD with the -p option to assign the port from the parameter file.

```
TACL> ASSIGN STDERR, $0
TACL> ADD DEFINE =TCPIP^PROCESS^NAME, FILES $ZTC1
TACL> RUN REPLICAT/IN GGSPARM.REPL1,OUT GGSRPT.REPL1/INITIALDATALOAD -p 12345
TACL> ASSIGN STDERR, $0
TACL> ADD DEFINE =TCPIP^PROCESS^NAME, FILES $ZTC2
TACL> RUN REPLICAT/IN GGSPARM.REPL2,OUT GGSRPT.REPL2/INITIALDATALOAD -p 12346
```

> **Note**
>
> Since the Replicats are started statically, they will not be restarted by Manager if there is a system problem.

Loading Oracle, Microsoft, or Sybase SQL Server Tables

NonStop tables and files can be synchronized with Oracle or SQL Server tables in very much the same way as NonStop-to-NonStop synchronization.

Loading to Oracle or SQL Server

To load to Oracle or SQL Server:

1. Run `DEFGEN` to export source data definitions to the target system. Be sure to satisfy any other prerequisites.
2. Start the Collector on the target system:

   For UNIX:

   ```
   $server –d /ggs/mydb.def 2> server.log &
   ```

   For Windows:

   ```
   server –d \ggs\mydb.def 2> server.log
   ```

   Use the –d option to specify the definitions file created by `DEFGEN` (`mydb.def`).
3. Create an Extract parameter file to perform initial table extract over TCP/IP.
4. Run the Extract program to extract the data into a flat file on the target system:

   ```
   TACL> RUN GGS.EXTRACT /IN GGSPARM.TRANINI, OUT $S.#TRAN/
   ```

   This command creates a flat file on the target. If you specified the `FORMATASCII, SQLLOADER` in a parameter file for Oracle, Oracle GoldenGate generates the flat file in a format that SQL*Loader can read. If you specified `FORMATASCII, BCP` in the parameter file for SQL Server, Oracle GoldenGate generates a flat file that is compatible with the BCP utility.
5. Create a Replicat parameter file using the `MAP` parameter to map between source and target tables.
6. Run Replicat to create files called `TRANSTAB.ctl` and `TRANSTAB.run` for Oracle, and `TRANSTAB.bat` and `TRANSTAB.fmt` for SQL Server. These files contain mapping definitions and run commands required to load.

   For UNIX:

   ```
   replicat paramfile/ggs/dirprm/tranini.prm
   ```

   For Windows:

   ```
   C:\replicat paramfile\ggs\dirprm\tranini
   ```
7. Load the data.

   For UNIX:

   ```
   $TRANSTAB.run
   ```

   For Windows:

   ```
   TRANSTAB.bat
   ```

Initial Sync Parameter File Examples

This sections contains these examples:

- An Extract parameter file example for Oracle running on UNIX
- A Replicat parameter file example for Oracle running on UNIX

- An Extract parameter file example for SQL Server running on Windows
- A Replicat parameter file example for SQL Server running on Windows

Sample NonStop to Oracle Parameter Files

Following are examples of NonStop to Oracle Extract parameter files.

Extract Parameter File GGSPARM.ORAINI:

```
SOURCEISFILE, FASTUNLOAD
FORMATASCII, SQLLOADER
RMTHOST ntbox12, MGRPORT 7809, PARAMS "-d c:\ggs\dirdef\source.def"
RMTFILE TRANSTAB.dat, PURGE
FILE \SRC.$D2.MYDB.TRANSTAB;
```

- `FORMATASCII, SQLLOADER` specifies the data format is compatible with Oracle's SQL*Loader utility.
- `RMTFILE` identifies `TRANSTAB.dat` as the source table.

Replicat Parameter File /ggs/dirprm/tranini.prm:

```
GENLOADFILES
USERID me, PASSWORD orapw
SOURCEDEFS /ggs/mydb.def
MAP $D2.MYDB.TRANSTAB, TARGET ORATRANS;
RMTFILE /ggsdat/tranini, PURGE
FILE \SRC.$D2.MYDB.TRANSTAB;
```

- `GENLOADFILES` generates load control files and then quits. These control files generate maps, even between dissimilar tables.
- `USERID` and `PASSWORD` specify the database log on.
- `SOURCEDEFS` specifies the location of the NonStop definitions exported by `DEFGEN` (These are required to generate a load map.)
- `MAP` specifies the source to target relationship of the NonStop to Oracle table.
- Errors are displayed to the screen and detailed messages are written to the `TRANSTAB.err` and `TRANSTAB.log` files

Sample SQL Server Parameter Files

Following are examples of parameter files for SQL Server.

Extract parameter file GGSPARM.SQLINI:

```
SOURCEISFILE, FASTUNLOAD
FORMATASCII, BCP
RMTHOST ntbox12, MGRPORT 7809, PARAMS "-d c:\ggs\dirdef\source.def"
RMTFILE C:\GGS\TRANSTAB.dat, PURGE
TABLE $DATA.MASTER.TRANS
```

- `FORMATASCII, BCP` specifies the data format is compatible with the Microsoft `BCP` utility.
- `RMTFILE` identifies `TRANSTAB.dat` as the source table. Using the `dat` extension makes it compatible with the load functions.

To load data to SQL Server, you must use the `BCP` template provided by Oracle GoldenGate. You can call `BCP` from your Replicat parameter file or run it interactively from the operating system shell. The template tells Replicat how data is laid out in the SQL Server target.

Replicat parameter file for GGSPARM.TRANINI

```
GENLOADFILES BCPFMT.TPLTARGETDB
TARGETDB MYAPP, USERID MYNAME, PASSWORD MSPW
SOURCEDEFS c:\ggs\mydb.def
MAP $D2.MYDB.TRANSTAB, TARGET SCHEMA.ORATRANS;
```

Limiting the Enscribe Source Key Range for Initial Load

If your parameters meet the requirements, the FILE parameter options STARTKEY and ENDKEY can be used to limit the range of Enscribe records selected for a SOURCEISFILE initial-load process. This allows you to load subsets of the data for different purposes or to break up the a large initial data load. Refer to *Reference for Oracle GoldenGate on HP NonStop Guardian* FILE | TABLE parameter for specifics on the requirements and how to use STARTKEY and ENDKEY.

Restarting an Initial Load

You can restart initial loads using the RESTARTCHECKPOINTS option of the SOURCEISFILE or SOURCEISTABLE parameter if your Extract is added from GGSCI.

You can use RESTARTCHECKPOINTS for:

- SQL/MP source tables with or without the SQLPREDICATE option
- Enscribe whether or not you use the FILE STARTKEY and ENDKEY options
- Both SQL/MP and Enscribe with or without FASTUNLOAD.

Refer to *Reference for Oracle GoldenGate on HP NonStop Guardian* for additional conditions and restrictions for using the SOURCEISFILE RESTARTCHECKPOINTS option.

The messages generated when the SOURCEISFILE Extract restarts vary based on the type of database and the parameters and options that are used. Some different types of examples are shown next.

Example 1 SQL/MP tables produced without using FASTUNLOAD
A message similar to the following is produced for SQL/MP source tables without FASTUNLOAD. In this example the option SQLPREDICATE is being used and WHERE (STATE = "CA") is the user's predicate. AC_KEY is the multi-column key for the restart.

```
Output extract file \NY.$DATA02.ACDAT.PA000009 Write Position: RBA 19126
Extract SourceIsFile process is restartable
Processing File \NY.$DATA02.ACDAT.ACCT
Using this SQL statement to retrieve data:
SELECT * FROM \NY.$DATA02.ACDAT.ACCT WHERE (STATE = "CA") AND AC_KEY1, AC_KEY2, AC_KEY3
> 13 ,4781 ,27 BROWSE ACCESS
```

Example 2 SQL/MP or Enscribe tables produced using FASTUNLOAD
A message similar to the following is produced for SQL/MP or Enscribe source tables using FASTUNLOAD. The restart key is RBA 9555968 of partition $DATA03.

```
Output extract file \NY.$DATA02.ACDAT.PA000009 Write Position: RBA 19126
Extract SourceIsFile process is restartable
Processing File \NY.$DATA02.ACDAT.ACCT2
Processing Partition \NY.$DATA03.ACDAT.ACCT2
Positioning Restart at RBA 9555968
```

Example 3 Enscribe tables produced without FASTUNLOAD or STARTKEY
A message similar to the following is produced for Enscribe that is not using FASTUNLOAD and without STARTKEY. The CUST-KEY used for the restart is 1234.

```
Output extract file \NY.$DATA02.ACDAT.PA000009 Write Position: RBA 19126
ExtractSourceIsFile process is restartable
Processing File \NY.$DATA02.ACDAT.ALTPART
Processing using restart values ( CUST-KEY = 1234 )
```

Example 4 Enscribe tables produced without FASTLOAD and with a STARTKEY
A message similar to the following is produced for Enscribe without FASTUNLOAD and with STARTKEY. The CUST-KEY used for the restart is 1234.

```
file $data02.acdat.altpart, startkey (CUST-key = 0000), def ens-rec,
                          endkey   (CUST-key = 5555);
file $data02.acdat.altpart, startkey (CUST-key = 5556), def ens-rec,
                          endkey   (CUST-key = 999999);
Output extract file \NY.$DATA02.ACDAT.PA000009 Write Position: RBA 19126
Extract SourceIsFile process is restartable
Processing File \NY.$DATA02.ACDAT.ALTPART
Processing using restart values ( CUST-KEY = 1234 )
Finished to EndKey ( CUST-KEY = 5555 )
Processing from StartKey ( CUST-KEY = 5556 )
Finished to EndKey ( CUST-KEY = 999999 )
```

Loading Initial Data from Windows and Unix

Use Replicat to load data from a Windows or UNIX system into a NonStop target database.

Integrating Source and Target Data

When only a subset of source rows or columns are needed in the target, you can use one of the following techniques to integrate selected data into your target:

- Selecting on the source with WHERE or FILTER
- Mapping columns on the target with COLMAP

When the data source is a SQL table, you can specify SQL Views. This allows automatic filtering of columns before transmission.

Data transformation (such as six-to-eight digit date conversion) takes a little extra effort during the load process. There are a couple of ways to achieve initial loads in this situation.

The first solution involves extracting the entire table into a flat file. In this case, do not specify FORMATASCII. Next use Replicat to load the table using the SPECIALRUN parameter. This method, while slower than native loads, is often sufficient and allows field conversion functions to be used during replication.

The second solution is to perform all data mapping on the NonStop before transmission on the target side. This means that all conversion work is performed by Extract. Using this strategy can result in less network traffic, since filtering can be performed before data reaches the pipe. However, this can also require the creation of a dummy table or DDL definition on the NonStop side that mimics the structure of the real target table.

Distributing Extracted Data

In addition to extracting and replicating database changes, Extract can forward and distribute changes that have already been extracted. This process is known as *data pumping*.

Use data pumping in the following scenarios:

- A network or target system may be down for an extended time, but extraction or logging activities must occur constantly.
- Data extracted by Logger must be forwarded over TCP/IP to non-NonStop systems.

Running Extract for these purposes is nearly identical to capturing data from TMF audit trails. To run Extract in this manner, perform the following tasks.

1. Using the `EXTTRAILSOURCE` or `LOGTRAILSOURCE` option, create an initial Extract checkpoint with the GGSCI `ADD EXTRACT` command.
2. Add a local or remote Oracle GoldenGate trail with the GGSCI `ADD EXTTRAIL` or `ADD RMTTRAIL` command. By adding the trails, you direct Extract where to write the data you need.
3. Set up an Extract parameter file.
4. Start Extract using the GGSCI `START EXTRACT` command.

Direct File Extraction

Rather than capturing from trails, you can extract directly from a file or a sequence of files. You can read a file directly only when the following conditions are true:

- The file or sequence of files is entry-sequenced.
- Only inserts occur against the files (no updates).
- Records are inserted only at the end of the file.

Use this feature when:

- An entry-sequenced, BASE24 TFL/PTLF, or Connex Advantage file
- The input files meet the conditions described above.
- You want to transfer the batch file contents a little at a time throughout the day ("trickle" transfer), rather than all at once at the end of the day.

To extract directly from a file:

1. Enter a GGSCI `ADD EXTRACT` command, specifying the `FILETYPE` parameter. `FILETYPE` indicates the type of file from which you are reading.
2. If more than one file in a sequence might be open at a time, start Extract for each file in use simultaneously. Enter an `ALTINPUT` parameter in each process's parameter file with a `RANGE` option to distribute the files among the processes.

Batch Processing

You can configure Extract and Replicat to run in batch when capturing and delivering incremental changes is not appropriate for the application. You can configure ongoing batch runs for a specific time daily, or special, one-time batch runs.

One-Time Database Extraction

You can run Extract against a specified period of audit trail or Oracle GoldenGate trail data a single time. Do this, for example, to extract changes to a particular account in a database over the course of a day.

To extract changes for a specific period, perform the following steps.

1. Set up a parameter file using the NonStop editor.
2. Use `SPECIALRUN` to capture data from TMF-audit trails. `SPECIALRUN` indicates that no checkpoints are maintained.
3. To extract data from an Oracle GoldenGate trail, use the `SPECIALRUN`, `EXTTRAILSOURCE` or `LOGTRAILSOURCE` option.
4. Set up `BEGIN` and `END` parameters to designate the period of activity to extract.
5. Designate an `EXTFILE` or `RMTFILE` rather than an extract trail. If you require multiple trails, add the `MAXFILE` argument to the `EXTFILE` or `RMTFILE` parameter.
6. Specify additional parameters as needed.
7. Start Extract from TACL, as in this example.

```
TACL> RUN EXTRACT /IN GGSPARM.SPECEXT, OUT GGSRPT.SPECEXT/
```

Trickle Batch Processing

When you are extracting batch files using `RMTBATCH`, you may need to perform the following steps:

1. Use the `SYSKEYCONVERT` parameter in the Extract parameter file if the input record's length is variable. This specifies the format of the `SYSKEY` in the output.
2. Use the `POSITIONFIRSTRECORD` parameter to reread an input file when you have used `SYSKEYCONVERT`. `POSITIONFIRSTRECORD` resets Extract to read from the beginning of the input file.

   RMTBATCH has the following dependencies:

   - `RMTBATCH` is only supported to a z/OS target using OGG 12.2
   - `RMTBATCH` must precede the names of files and tables containing data you want to write to the remote file.
   - Precede the `RMTBATCH` entry by a `RMTHOST` entry to identify the target computer and TCP/IP port.

Determining the Next File

Use `ALTINPUT` for direct file extraction. With ACI files, multiple files can be in use at one time. For example, processing can continue for Monday's file after midnight, while Tuesday's file is opened for new data. To handle a multiple file situation, run more than one Extract process for the file sequence. Use the `ALTINPUT RANGE` option to distribute the files across the processes so that Extract never processes two files in sequence. You can also use `ALTINPUT` to specify the access mode of the file open, and to move Extract to the next sequential file if an application has a file open that it is not updating.

By default, Extract looks for the next alphabetical file name. The file name must conform to a template for the file type, which defaults to predefined characteristics. You can also specify the template by parameter.

If the file type is `ACITLF` or `ACIPTLF`, then the template is in the form `$VOL.SUBVOL.XXYYMMDD`, where *XX* is a two character prefix, followed by a six digit date.

If the file type is `ACITLFX` or `ACIPTLFX`, the template is in the form `$VOL.SUBVOL.XMMDDNNN`, where `X` is a one character prefix, followed by a month, day and three digit sequence number.

When specifying any of the ACI file types in the `FILETYPE` option, do not include the date or sequence number. The next file is the one following the current file in name order, and must also satisfy any `RANGE` criteria in the `ALTINPUT` parameter.

If the file type is `ENTRY`, you specify the template in the `ALTINPUT` parameter `TEMPLATE` option. NonStop wildcards are acceptable. For example, the template `$DATA*.MYDAT.FL*` processes files starting with `FL` residing on different `$Data` volumes.

When using `FILETYPE ADVANTAGE`, specify the first file to process, not the file prefix. By default, the next file is the next file name to fit the template. As an alternative, you can use `ALTINPUT USENEXTMODIFIED`. This option selects the next file modified after the current file that also fits the template.

When the Next File is Processed

Before moving to the next file in a sequence, Extract must process the entire contents of the current file. By default, Extract uses the following rules to determine that the current file has been exhausted and the next file is ready for processing.

- End-of-file was reached in current file at least five seconds earlier, and no new data has appeared since.
- No processes have the current file open for write access.
- The next file exists and has at least one record in it.
- The next file was modified after the current file.

You can modify these rules with the `NOWAITNEXTMODIFIED`, `WAITNEXTRBA`, and `OPENTIMEOUT` options for the `ALTINPUT` parameter.

4

Configuring Oracle GoldenGate Security

This topic discusses the security features that you can use to protect your Oracle GoldenGate for HP NonStop environment as well as the data that is being processed.

This topic includes the following sections:

The following tables summarizes the security features that are available in Oracle GoldenGate for HP NonStop:

Table 4-1 Oracle GoldenGate Security Features

Security feature	Description
Extract and Logger trail	Extract and Logger trail files write locations are restricted from the Oracle GoldenGate Install `subvol`, `$SYSTEM`, and all `TMF auditvol` drives.
Encryption	Options are available for encrypting and decrypting: • data in an extract file or trail • database passwords • data sent across TCP/IP
Command security	Sets user-level permissions for accessing Oracle GoldenGate commands through GGSCI.

Using Encryption

This section contains instructions for encrypting and decrypting the following:

- The trail or extract file that holds data being processed by Oracle GoldenGate
- A database password
- The data sent across TCP/IP

How Data is Encrypted?

The following encryption methods are available:

- **Advanced Encryption Standard**: AES is a Symmetric-key encryption using a block cipher with a fixed block size of 128 bits. The Oracle GoldenGate implementation supports all three key sizes of 128, 192, or 256 bits.
- **Encrypting the database password**: AES encryption is available to encrypt passwords.
- **Encrypting data sent across TCP/IP**: AES encryption methods are available to encrypt data over TCP/IP.
- **Encrypting trails or extract files**.
  AES encryption methods are available to encrypt trails or extract files.

Encrypting Trail or Extract Files

You can encrypt the data in any local or remote trail or file.

> **Note**
>
> This feature cannot be used when FORMATASCII is used to write data to a file in ASCII format. The trail or file must be written in default canonical format.

To encrypt trail or extract files

1. In the Extract parameter file, list the following parameter before all trails or files that you want to be encrypted. You can list multiple trails or files after one instance of this creating trails all with the same method of AES and KEYNAME.

   ```
   ENCRYPTTRAIL (<AES>, KEYNAME <name>)
   ```

2. To disable encryption for any files or trails listed in the Extract parameter file, precede their entries with the following parameter:

   ```
   NOENCRYPTTRAIL
   ```

3. In the Replicat parameter file, include the following parameter so that Replicat decrypts the data for processing.

   ```
   DECRYPTTRAIL(<AES>, KEYNAME <name>)
   ```

   You also can use DECRYPTTRAIL for an Extract data pump to decrypt the data for column mapping, filtering, transformation, and so forth. You can then leave it decrypted for downstream trails or files, or you can use ENCRYPTTRAIL to encrypt the data again before it is written to those files. For more information on ENCRYPTTRAIL and DECRYPTTRAIL, see *Reference Guide for Oracle GoldenGate for HP NonStop*.

Encrypting a Database Password

Use the following steps to encrypt the database password used by the Oracle GoldenGate processes.

1. Run GGSCI and issue the ENCRYPT PASSWORD command to generate an encrypted password. The command provides the following options.

   - The default ENCRYPT PASSWORD command, without any options, generates an encrypted password using a default key that is randomly generated by Oracle GoldenGate.

     ```
     ENCRYPT PASSWORD password
     ```

   - ENCRYPT PASSWORD with the KEYNAME *keyname* generates an encrypted password using a user-defined key contained in the ENCKEYS lookup file.

     ```
     ENCRYPT PASSWORD password KEYNAME keyname
     ```

     For *keyname*, specify the logical name for the key you want to use, as it appears in the local ENCKEYS file. To use this option, you must first generate a key, create an ENCKEYS file on the local system, and create an entry in the file for the generated key. For instructions, see "Generating Encryption Keys".

   The encrypted password is displayed to the screen when you run the ENCRYPT PASSWORD command.

2. Copy the encrypted password and paste it into the Oracle GoldenGate parameter file as follows.

```
USERID user_id, PASSWORD password, &
[KEYNAME {DEFAULT | keyname}]
```

Where:

- *user_id* is the database user name for the Oracle GoldenGate process.
- *password* is the encrypted password that is copied from the ENCRYPT PASSWORD command results.
- KEYNAME DEFAULT is required if the password was encrypted using ENCRYPT PASSWORD without the ENCRYPTKEY option.
- KEYNAME *keyname* is required if the password was encrypted using ENCRYPT PASSWORD with the KEYNAME *keyname* option. Specify the logical name of the key as it appears in the ENCKEYS lookup file.

Encrypting Data Sent Across TCP/IP

You can encrypt captured data before Oracle GoldenGate sends it across the TCP/IP network to the target system. On the target system, Oracle GoldenGate decrypts the data before writing it to the Oracle GoldenGate trails (unless trail encryption also is specified). By default, data sent across a network is not encrypted.

To encrypt data sent across TCP/IP

1. On the source system, generate one or more encryption keys and create an ENCKEYS file. See "Generating Encryption Keys" for more information.
2. Copy the finished ENCKEYS file to the Oracle GoldenGate installation location on all target systems. The key names and values in the source ENCKEYS file must match those of the target ENCKEYS file, or else the data exchange will fail and Extract and Collector will abort with the following message:

   ```
   GGS error 118 – TCP/IP Server with invalid data.
   ```

3. In the Extract parameter file, use the ENCRYPT option of the RMTHOST parameter to specify the type of encryption and the logical key name as shown:

   ```
   RMTHOST hostname, MGRPORT port, ENCRYPT <AES>, KEYNAME keyname
   ```

4. If using a static Collector and AES encryption, append the following additional parameters in the Collector startup string:

   ```
   -KEYNAME <name>
   -ENCRYPT <AES>
   ```

   Where:

   - KEYNAME *name* specifies the name of the key.
   - ENCRYPT <*AES*> specifies AES encryption.

   An example of encrypting data sent across TCP/IP:

   ```
   RMTHOST sys1, MGRPORT 7840, ENCRYPT AES128, KEYNAME superkey
   ```

   Collector matches these parameters to those specified with the KEYNAME and ENCRYPT options of RMTHOST.

Generating Encryption Keys

You must create at least one encryption key and two ENCKEYS lookup files, one on the source and one on the target, if you want to:

- Encrypt data sent across TCP/IP
- Use a user-defined key to encrypt the database password

This procedure is *not required* if:

- You are using a default key to encrypt the database password.
- You are encrypting a trail or extract file.

You can define your own key or run the Oracle GoldenGate KEYGEN utility to create a key randomly.

To define your own key

- The key name can be a string of 1 to 24 alphanumeric characters without spaces or quotation marks.
- The key value can be up to 128 bits (16 bytes) as a quoted alphanumeric string (for example "Dailykey") or a hex string with the prefix 0x (for example 0x420E61BE7002D63560929CCA17A4E1FB).

To Use KEYGEN to Generate a Key

Change subvolumes to the Oracle GoldenGate installation location on the source system, and issue the following shell command. You can create multiple keys, if needed. The key values are returned to your screen.

```
TACL> RUN KEYGEN key_length number
```

Where:

- *key_length* is the encryption key length, up to 128 bits.
- *number* represents the number of keys to generate.

To store the keys for use by Oracle GoldenGate

1. On the source system, open a new ASCII text file.

   For each key that you generated, enter a logical name followed by the key value itself. Place multiple key definitions on separate lines. Do not enclose a key name or value within quotation marks; otherwise it will be interpreted as text. Use the following sample file as a guide.

   ```
   ## Key name            Key Value
   superkey               0x420E61BE7002D63560929CCA17A4E1FB
   secretkey              0x027742185BBF232D7C664A5E1A76B040
   superkey1              0x42DACD1B0E94539763C6699D3AE8E200
   superkey2              0x0343AD757A50A08E7F9A17313DBAB045
   superkey3              0x43AC8DCE660CED861B6DC4C6408C7E8A
   ```

2. Save the file as ENCKEYS without an extension in the Oracle GoldenGate installation location. The name must be in upper case.

3. Copy the ENCKEYS file to the target Oracle GoldenGate installation location. The key names and values in the source ENCKEYS file must match those of the target ENCKEYS file, or else the data exchange will fail and Extract and Collector will abort with the following message:

   ```
   GGS error 118 – TCP/IP Server with invalid data.
   ```

Example 4-1 Using KEYGEN to Generate a Key

```
TACL> RUN KEYGEN 128 4
```

Using Command Security

You can establish command security for Oracle GoldenGate to control which users have access to which Oracle GoldenGate functions. For example, you can allow certain users to issue INFO and STATUS commands, while preventing their use of START and STOP commands. Security levels are defined by the operating system's user groups.

To implement security for Oracle GoldenGate commands, you create a CMDSEC file in the Oracle GoldenGate installation location. Without this file, access to all Oracle GoldenGate commands is granted to all users.

To implement command security

1. Open a new ASCII text file.
2. Referring to the following syntax and the example on "Table 4-2", create one or more security rules for each command that you want to restrict, one rule per line. Order the rules from the most specific (those with no wildcards) to the least specific. Security rules are processed from the top of the CMDSEC file downward. The first rule satisfied is the one that determines whether access is allowed.

   Separate each of the following components with spaces or tabs.

   ```
   command_name command_object user_group user YES|NO
   ```

   Where:

   - *command_name* is a GGSCI command name or a wildcard, for example START or STOP or *. Command names are not validated for accuracy.
   - *command_object* is any GGSCI command object or a wildcard, for example EXTRACT or REPLICAT or MANAGER. Command objects are not validated for accuracy.
   - *user_group* is the numeric ID of the Guardian user group, such as 100 or 255. You can use a wildcard to specify all groups.
   - *user* is the Guardian user numeric ID, such as 2 or 255. You can use a wildcard to specify all users.
   - *YES|NO* specifies whether access to the command is granted or prohibited.
3. Save the file as CMDSEC in the Oracle GoldenGate installation location

The following example illustrates the correct implementation of a CMDSEC file on a NonStop system.

Table 4-2 Sample Cmdsec File with Explanations

File Contents	Explanation
`--GG command security`	Comment line

Table 4-2 (Cont.) Sample Cmdsec File with Explanations

File Contents	Explanation
`STATUS REPLICAT 100 15 NO`	`STATUS REPLICAT` is denied to user `15` of group `100`.
`STATUS * 100 * YES`	Except for the preceding rule, all users in `100` are granted all `STATUS` commands.
`START REPLICAT 255 * YES`	`START REPLICAT` is granted to all members of the `Super` (255) group.
`START REPLICAT * * NO`	Except for the preceding rule, `START REPLICAT` is denied to all users.
`* EXTRACT 200 * NO`	All `EXTRACT` commands are denied to all groups with ID of 200.
`* * 255 255 YES`	Grants the `Super.Super` user any command.
`* * * * NO`	Denies all commands to all users. This line covers security for any other users that were not explicitly granted or denied access by preceding rules. Without it, all commands would be granted to all users except for preceding explicit grants or denials.

Table 4-3 Incorrect CMDSEC Entries

File Contents	Description
`STATUS REPLICAT 100 15 NO`	`STATUS REPLICAT` is denied to user `15` of group `100.`
`STOP * 100 * NO`	All `STOP` commands are denied to everyone in group `100`.
`STOP * * 15 YES`	All `STOP` commands are granted to user `15`.

The above *incorrect* example illustrates what to avoid when creating a `CMDSEC` file. The order of the entries in Table 4-3 causes a logical error. From the first rule (line 1), you can see that user `15` is a member of group `100`. The second rule (line 2) denies all `STOP` commands to all members of group `100`. The third rule (line 3) grants all `STOP` commands to user `15`. However, because `15` is a member of the `100` group, he has been denied access to all `STOP` commands by the second rule.

The proper way to configure this security rule is to set the user-specific rule before the more general rules. Thus, to correct the error, you would reverse the order of the two `STOP` rules.

Securing the CMDSEC File

Because the `CMDSEC` file is a source of security, it must be secured. You can grant read access as needed, but Oracle GoldenGate recommends denying write and delete access to everyone but the Oracle GoldenGate administrator. For example, a proper security string might be `"NUUU"`.

5

Configuring the Manager and Collector

The Manager and the Collector are Oracle GoldenGate components that facilitate day-to-day data management. See how to configure, start, and run these components.
This topic includes the following sections:

Introducing Manager

Manager runs as a NonStop process, ensuring that Oracle GoldenGate for NonStop components run and restart if a CPU failure or operator error occurs. Manager spawns a copy of itself so that tasks that take longer, such as duplicating a TMF audit trail, do not interfere with real-time tasks. Tasks are divided between the Manager and its child process accordingly. Manager tasks on NonStop include:

- Starting Logger, Extract, Replicat and Syncfile
- Monitoring and reporting status of Oracle GoldenGate processing
- Starting the dynamic Collector process on the target
- Automatically restarting critical processes
- Threshold reporting, such as when Extract falls behind the TMF-audit trail
- Managing resources for the TMF audit trail, such as maintaining copies of audit trails on backup volumes
- Purging trails when Extract and Replicat has finished with them
- Pre-allocating log trail space for Logger processing

Configuring and Starting Manager

Now that you have your GGSCI prompt, you are ready to configure and start Manager. To configure Manager, create an appropriate parameter file. To control an Oracle GoldenGate Manager process, use the following commands.

Command	Description
`START MANAGER`	Starts Manager.
`STOP MANAGER`	Stops Manager. You can stop Manager gracefully, or forcefully with the ! option.

Creating and Configuring the Manager Parameter File

Enter Manager parameters in the `MGRPARM` file. If no `MGRPARM` file exists, default management parameters are taken. To add parameters, edit this file using the `EDIT PARAMS MGRPARM` command.

Manager retrieves parameters as established by GGSCI `ATCONFIG` commands. These parameters affect audit trail resource management.

See Oracle GoldenGate Parameters for more information about Manager parameters.

A Sample Manager Parameter File

A Manager parameter file would be similar to this sample.

Example 5-1 Sample Manager Parameter File

```
TCPIPPROCESSNAME $ZTC2
PORT 7844
DYNAMICPORTLIST 7850 - 7880, 7895
CHECKMINUTES 30
PURGEOLDEXTRACTS $DATA1.GGSDAT.*, USECHECKPOINTS, MINKEEPDAYS 2
THRESHOLD 30
LAGREPORTMINUTES 60
LAGINFOMINUTES 10
LAGCRITICALMINUTES 10
LOGFILESBEHIND 2
LOGFILESBEHINDINFO 10
DOWNCRITICAL
DOWNREPORTHOURS 1
```

In this Sample Manager Parameter File

- `TCPIPPROCESSNAME` specifies the TCP/IP process. The default process is $ZTC0. Use the `TCPIPPROCESSNAME` parameter to specify a process other than the default.
- Specify the `PORT` parameter so Manager can create a dynamic Collector process.
- `DYNAMICPORTLIST` specifies up to 256 entries for ports to be dynamically assigned to processes started by Manager. If no dynamic ports are specified, Manager starts with port 7819 and increments until it finds an available port.
- `CHECKMINUTES 30` directs Manager to perform maintenance activities every 30 minutes. The default is 10 minutes.
- Use the `PURGEOLDEXTRACTS` parameter when multiple Replicat processes are reading a set of trails. For this sample, `PURGEOLDEXTRACTS` directs manager to purge old files from the trail `$DATA1.GGSDAT.*`. The options:
  - `USECHECKPOINTS` specifies that Replicat checkpoints are to be used to determine when Replicat has finished processing.
  - `MINKEEPDAYS 2` purges files only after they have been closed for 2 days.
- `THRESHOLD 30` directs Manager to generate an event message when the number of audit files remaining to be processed falls below 30%.
- `LAGREPORTMINUTES 60` specifies that Manager check lag every 60 minutes.
- `LAGINFOMINUTES 10` specifies that Manager report lag information to the event log every 10 minutes.
- `LAGCRITICALMINUTES 10` specifies that Manager write a critical message to the event log when there is 10 minute lag.
- `LOGFILESBEHIND 2` sends a critical message whenever a process lags a specified number of files behind the current log trail file.
- `LOGFILESBEHINDINFO 10` sends an informational message whenever the process falls the specified number of files behind.
- `DOWNCRITICAL` sends a critical message whenever Extract or Replicat abends.
- `DOWNREPORTHOURS` sends reports of Extract and Replicat abending.

Starting and Stopping Manager

You must start Manager before you can configure and run other Oracle GoldenGate components. The following example starts Manager in `CPU 3`. Manager selects a CPU in which to run a backup process for fault-tolerance.

```
GGSCI> START Manager, CPU 3, PRI 170
```

If Manager encounters a TCP/IP error, for example if it attempts to bind to a port that is in use, it retries the error every 60 seconds and does not abend after a set number of attempts. Unlike other Oracle GoldenGate processes, it does not use the `TCPERRS` file to set the delay and the number of retries.

Manager runs indefinitely, or until you enter the GGSCI `STOP MANAGER` command. You might stop Manager if you need to stop the Extract and Replicat groups it manages or if you to want to activate a change to a Manager parameter.

See STOP MANAGER for more information about GGSCI commands for Manager.

Configuring and Running the Collector

The Collector collects data from Extract and writes data to files on the target system. Extract requests Manager to start a collector process when it sees data must transmit over TCP/IP to a remote trail. Once started, the Collector waits for and performs requests to write, open, and close files in the Oracle GoldenGate trail during Extract processing.

> **Note**
>
> You do not need to run collector if data transmits over an Expand connection.

Maintaining Ports for Remote Connections through Firewalls

If a firewall is being used at an Oracle GoldenGate target location, additional ports are required on the target system to receive dynamic TCP/IP communications from remote Oracle GoldenGate processes. These ports are:

- One port for *each* Collector process that is started by the local Manager to receive propagated transaction data from remote online Extract processes. When an Extract process sends data to a target, the Manager on the target starts a dedicated Collector process.
- One port for each Replicat process that is started by the local Manager as part of a remote task. A remote task is used for initial loads and is specified with the `RMTTASK` parameter. This port is used to receive incoming requests from the remote Extract process. For more information see, RMTTASK
- Some extra ports in case they are needed for expansion of the local Oracle GoldenGate configuration.
- Ports for the other Oracle GoldenGate products if they interact with the local Oracle GoldenGate instance, as stated in the documentation of those products.

To specify these ports, use the `DYNAMICPORTLIST` parameter in the Manager parameter file. Follow these guidelines:

- You can specify up to 300 ports in any combination of the following formats:

```
7830, 7833, 7835
7830-7835
7830-7835, 7839
```

- The ports must be unreserved and unrestricted.
- Each Manager instance on a system must use a different port list.

Although not a required parameter, DYNAMICPORTLIST is strongly recommended for best performance. The Collector process is responsible for finding and binding to an available port, and having a known list of qualified ports speeds this process. In the absence of DYNAMICPORTLIST (or if not enough ports are specified with it), Collector tries to use port 7840 for remote requests. If 7840 is not available, Collector increments by one until it finds an available port. This can delay the acceptance of the remote request. If Collector runs out of ports in the DYNAMICPORTLIST list, the following occurs:

- Manager reports an error in its process report and in the Oracle GoldenGate LOGGGS.
- Collector retries based on the rules in the Oracle GoldenGate tcperrs file. For more information about the tcperrs file, see section"TCP/IP Error Handling ".

For more information see, DYNAMICPORTLIST.

Configuring Collector

To configure a Collector, you must know the port the Collector will use, the host name or IP address where the remote trail resides, and edit your Manager and Extract parameter files. You may also specify a variety of operating options, described in the "Configuration Examples".

To configure and start Collector:

1. In the Manager parameter file, specify the port parameter, such as: PORT 7809.
2. In the Extract parameter file, specify the RMTHOST parameter as follows:

```
RMTHOST host, [MGRPORT port_number] [, option] [, . . .]
```

Argument	Description
host	Either a remote system name or an IP address, such as: RMTHOST eastnode or RMTHOST 192.0.2.20.
MGRPORT *port_number*	Specify the port that is defined in the Manager parameter file.
options	You can specify a variety of options, including Collector parameters. See "Creating and Configuring the Manager Parameter File" for information on these options. See Oracle GoldenGate Parameters for information about other RMTHOST options.

3. The remote system must be the same system on which Collector was started, and the port number must match the port number in the Collector startup command. See "Configuration Examples" for more information.
4. If you specify a remote system name in the RMTHOST parameter, you must also enter the remote system name in the TCP/IP hosts file on the target system, or on the names server for your network. For example, if you specify: RMTHOST eastnode, you must make an entry similar to: 192.0.2.20 eastnode in the HOSTS file.

If you specify the remote system IP address in the RMTHOST parameter, there is no need to make a corresponding HOSTS file entry.

Configuration Examples

Following are the examples for configuration:

To configure Collector for port 5432 on remote system named eastnode:

1. In the Manager parameter file, specify: `PORT 5432`
2. In the Extract parameter file, specify: `RMTHOST eastnode, MGRPORT 5432,` and options if desired.
3. In the TCP/IP `HOSTS` file, enter: `192.0.2.20 eastnode`

To configure Collector for the default port on remote system address 192.0.2.20:

1. In the Manager parameter file, specify: `PORT 7809`
2. In the Extract parameter file, specify: `RMTHOST 192.0.2.20,` and options if desired.
3. No TCP/IP hosts file entry is required.

The TCP/IP Port

There are two ways to use Collector and your TCP/IP port: dynamically and explicitly. The dynamic method lets Extract request Manager to start Collector as needed. However, a user can explicitly start the Collector and let it run in the background, ready to transmit data as needed. This method is called the *explicit method*.

Dynamic Method

Dynamic method is the default way to use Collector. The examples above illustrate how this is configured: a port is specified in the Manager parameter file, a remote trail is specified in the Extract parameter file, and, if required, the IP address is added to your hosts file.

Explicit Method

When capturing data over TCP/IP to remote systems that do not support dynamic Collectors, you must explicitly start a Collector on the target system before starting Extract. Each Extract must explicitly name the port to which it is writing, using the RMTHOST parameter.

To explicitly configure your Collector, start GGSCI and enter the following:

```
TACL > ASSIGN STDERR, event_message_collector
TACL > RUN SERVER /NOWAIT/ [-P port_number]
```

> **Note**
>
> Your *event_message_collector* may be the standard system log, $0, or a virtual process, such as $VHS.

In the above example, the Collector listens on port `12345`. When you start the Collector, it references a default TCP/IP process (`$ZTC0`). You can change this to the process of your choice by running a `DEFINE` statement before you start your collector.

```
TACL > ASSIGN STDERR, $0
TACL> DEFINE =TCPIP^PROCESS^NAME, FILE $ZTC8
TACL > RUN SERVER /NOWAIT, NAME $COLL/ -P 12345
```

See Collector Parameters for more information about the Collector parameters.

Example 5-2 Explicit Method

```
TACL > ASSIGN STDERR, $0
TACL > RUN SERVER /NOWAIT, NAME $COLL/ -P 12345
```

Monitoring Collector

Collector event messages are output to the `ggserr.log` file. You can view this file using the GGSCI `VIEW GGSEVT` command.

Security Considerations

The user ID under which the Collector is started determines whether target files can be written and purged. Ensure that the ID has the proper system access to the files and locations written by the Collector.

Collecting Between Open Systems and NonStop

Event messages created by the Collector and Replicat on Windows and UNIX systems can be extracted and sent back to EMS on NonStop systems. This feature enables centralized viewing of Oracle GoldenGate messages across platforms.

To collect events from other systems:

1. Run Collector on NonStop to collect and distribute EMS messages. For each EMSCLNT process, run one Collector process.

   The following example runs Collector and outputs its messages to $0.

   ```
   TACL> ASSIGN STDERR, $0
   TACL> RUN SERVER /NOWAIT/ -p 7880
   ```

2. Run the EMSCLNT utility on the remote target. EMSCLNT reads a designated error log and runs indefinitely, waiting for more messages to send. When EMSCLNT receives a message, it sends the message to a collector process on NonStop.

   See the examples for running EMSCLNT on open systems for syntax information.

   This UNIX example reads the file `ggslog.err` for error messages. Error messages are sent to the collector to the NonStop at IP address `192.0.2.2` listening on port `7850`. The Collector on NonStop writes formatted messages to EMS Collector `$0`.

The Windows example below (from the DOS prompt) reads the file `d:\ggserrs\log.txt` for error messages. Error messages are sent to the collector on host `ggs2` listening on port `9876`. The Collector on NonStop writes formatted messages to EMS Collector $P0.

```
> emsclnt -h ggs2 -p 9876 -f d:\ggserrs\log.txt -c $P0
```

Argument	Description
`-h ggs2`	The node on which the collector is being run. Can be a name or IP address. This is a required parameter.

Argument	Description
`-p 9876`	The port where the collector is listening for messages. This is a required parameter.
`-f d:\ggserrs\log.txt`	The error file where EMSCLNT retrieves error messages. This is a required parameter.
`-c $P0`	The collector where EMS messages should be written on the NonStop (default is $0).

Example 5-3 Running EMSCLNT on Open Systems

```
> $emsclnt -h 192.0.2.2 -p 7850 -f ggserr.log -c $0
```

6

Configuring Change Synchronization

Online change synchronization extracts data and transmits it to a target. Here you see how to prepare Extract, Replicat, and Logger, and how to work with parameter files.

This topic includes the following:

Introduction

If your application is TMF-protected, perform change synchronization using Extract and Replicat. Non-TMF applications use Logger and Replicat to perform the same functions. You can configure Oracle GoldenGate to process changes from the following sources:

- A TMF audit trail
- An intermediate Oracle GoldenGate trail, created either by Logger or Extract
- Directly from entry-sequenced files, or from BASE24, `TLF` or `PTLF` files.
- Oracle GoldenGate logs generated from non-TMF applications

Change Synchronization for TMF Applications

TMF application change synchronization requires at least one Extract group, one trail, and one Replicat group.

Configuring Extract

To configure and run Extract, you must create an Extract group and an Extract parameter file.

1. Start GGSCI:

   ```
   TACL> RUN GGSCI
   ```

2. Add an Extract group. Specify CPU and priority:

   ```
   GGSCI> ADD EXTRACT EXTORD, BEGIN NOW, CPU 1, PRI 160
   ```

3. Create the Extract parameter file:

   ```
   GGSCI> EDIT PARAMS EXTORD
   ```

Example 6-1 Sample Extract Parameter File

```
-- Extract parameter file for
-- TCUSTMER and TCUSTORD changes
EXTRACT EXTORD
GETROLLBACKS
EXTTRAIL \LA.$DATA03.JDSDAT.ET
TABLE $DATA11.JDSSOU.TCUSTMER;
TABLE $DATA11.JDSSOU.TCUSTORD;
```

The name of the parameter file is usually the same as the process group name. For more information on parameter guidelines, see "Working with Parameter Files".

Configuring Trails

Based on considerations such as performance, hard disk constraints, and data throughput speed, you can specify where you want your trails to reside. For example, if you are concerned about disk space in your Extract environment, you may choose to create your trails on the system where Replicat is installed.

To configure your trail, you must create an empty trail using GGSCI, then start its associated process. You can test the trail by checking to see data is being written to it. If an `INFO ALL` command shows data being written to your trail, it is configured correctly.

To add an Oracle GoldenGate trail:

1. Determine if your trail will run locally or remotely. Base this decision on performance considerations vs. data throughput speed.
2. If you are not at the GGSCI prompt, start GGSCI.

   ```
   TACL> RUN GGSCI
   ```

3. Add your trail using the following commands:

   ```
   GGSCI> ADD EXTTRAIL trail_path, EXTRACT extract_group, [trail_size_limit],
   [limit_of_files_waiting_to_be_written]
   GGSCI> ADD RMTTRAIL trail_path, EXTRACT extract_group, [trail_size_limit]
   ```

   For example, a local trail would read:

   ```
   GGSCI> ADD EXTTRAIL \LA.$DATA03.JDSDAT.ET, EXTRACT EXTORD, MEGABYTES 5, MAXFILES 10
   ```

To test a trail:

1. Issue the GGSCI command `INFO ALL`.
2. Check to make sure Extract and Replicat are both running, and checkpoint sizes show relative byte addresses.

Configuring Replicat

Replicat gathers data from your trail and delivers it to your target. A Replicat group contains the named Replicat itself, a Replicat parameter file, and checkpoint groups, as desired.

To configure Replicat:

1. From GGSCI, create a Replicat group:

   ```
   GGSCI> ADD REPLICAT group_name, EXTTRAIL trail_name
   ```

   For example:

   ```
   GGSCI> ADD REPLICAT REPORD, EXTTRAIL $DATA03.JDSDAT.ET
   ```

2. Launch a NonStop text editor to create a Replicat parameter file (or use GGSCI):

   ```
   TACL> TEDIT PARAMS REPORD
   ```

3. Enter your parameters as desired.

   The name of the parameter file is usually the same as the process group name. For more information on parameter guidelines, see "Working with Parameter Files". Following is a sample Replicat parameter file.

```
-- Replicat parameter file for replicating
-- TCUSTMER and TCUSTORD changes
REPLICAT REPORD
HANDLECOLLISIONS
PURGEOLDEXTRACTS
ASSUMETARGETDEFS
DISCARDFILE \LA.$DATA11.GGSDISC.REPORD, PURGE
MAP \LA.$DATA11.GGSSOU.TCUSTORD,
TARGET \NY.$DATA03.GGSTAR.TCUSTORD;
MAP \LA.$DATA11.GGSSOU.TCUSTMER,TARGET \NY.$DATA03.GGSTAR.TCUSTMER;
```

4. Start GGSCI, then start the Replicat you just configured.

```
GGSCI> START REPLICAT REPORD
```

5. Verify that Replicat is working and receiving data from Extract.

Change Synchronization for Non-TMF Applications

The Logger program, with the GGSLIB run-time library, extracts changed records from files that are not protected by TMF. A Logger process records database updates in a log trail, which feeds data to Replicat. Each Logger process in a set is named `$GGLnn`, where `nn` is a sequenced identifier beginning with `00`. For example, if you configure two Logger processes, they are named `$GGL00` and `$GGL01`. Each Logger group has one or more Logger process, a parameter file, one or more log trails, and file extraction lists.

Log trails are sets of files, written to disk, that hold data extracted and sent to a particular Logger. Each log trail is owned by one Logger process. The parameter file holds specific volume locations and the number and size of each log file.

A log trail's name is comprised of the volume and subvolume where Oracle GoldenGate Logger is installed, the process prefix, and a series of letters and numbers that grow depending on the number of log trails. For example, `$DATA1.GLOGGGL.AA000000` means Oracle GoldenGate Logger is installed on volume `Data1`, subvolume `GLOG`, the process prefix is `GGL`, and the trail itself is called `AA000000`.

To configure and run change synchronization for non-TMF applications, you must:

1. Create the `LOGPARM` parameter file.
2. Configure Logger and GGSLIB with the `ADD LOGGER` command.
3. Start Logger.
4. Bind GGSLIB to the non-TMF application.

Creating the LOGPARM File

Just as Extract and Replicat are controlled by parameter files, so is Logger. Unlike either Extract or Replicat, you must create the `LOGPARM` before you add your Logger to Oracle GoldenGate Manager.

To create a Logger parameter file:

1. Start GGSCI.

```
TACL> VOLUME Oracle_GoldenGate_installation_location
TACL> RUN GGSCI
```

2. Enter Logger and GGSLIB parameters into the `LOGPARM` file.

```
GGSCI> TEDIT PARAMS LOGPARM
```

3. Add required parameters, including:
   - LOG entries, where each entry describes both a process and a log trail in which the process records database updates. Each log process writes to exactly one log trail.
   - FILE parameters, which specify one or more files to be extracted by the current log process (The current log is the first log entry preceding the FILE entry.) The FILE entry may be a wildcard. In addition, FILE may specify compression of update records.
   - EXCLUDEFILE parameters, which specifically exclude a file from a list for a particular log if it has been included with FILE.
   - The primary and backup CPUs in which the particular log process will run.
   - The priority at which the Logger process will run (PRIORITY).

Logger parameters are detailed in *Reference for Oracle GoldenGate on HP NonStop Guardian*. The following section outlines a sample Logger parameter file.

Sample LOGPARM File

This sample parameter file configures two log processes $GGL00 and $GGL01. These process names have been explicitly set with the PROCESS parameter, but when not set the names default to $GGLnn. The system will increment nn from 00 so the default will generate the same process names in this instance.

> **Note**
>
> Parameter position is important. As soon as a log entry is specified with the log parameter, it becomes the current log. Parameters entered below the current log parameter apply only to the current log. For instance, in the following example, all parameters after the LOG $D3.GGSLOG.AA and before the LOG $D15.GGSLOG.BB entry apply to LOG $D3.GGSLOG.AA.

- Creates a log trail $D3.GGSLOG.AA that contains 10 files each sized at 500 megabytes (for a total of 5,000 megabytes). The file names will be AA000000, AA000001, through AA000009. As new files are required, the oldest one is recycled and takes the next sequence number; in this case, AA000000 will become AA000010. File space is pre-allocated by the GGSCI and Manager processes.
- Configures $GGL00 to run on CPU 9, with backup CPU 4.
- Specifies that data written by the application in $DATA4 will be logged to the log trail $D3.GGSLOG.AA.
- Specifies that data written by the application in $DATA5 will be logged to the log trail $D3.GGSLOG.AA.
- Excludes $DATA4.REPORTS.* from being logged to AA.
- Excludes $DATA4.DAT.TRANSFL from being logged to AA.

Logger GGL01:

- Creates a log trail $D15.GGSLOG.BB that contains 100 files each sized at 10 megabytes (for a total of 1,000 megabytes). The file names will be BB000000, BB000001, through BB000009. These files are recycled when needed.
- Configures $GGL01 to run in CPU 3, with backup CPU 2.

- Specifies that data written by the application to files in any location should be written to the BB log trail, except $DATA4.REPORTS.* and any data already captured by $GGL00 (in this case, $DATA4.*.* and $DATA5.*.*). $DATA4.DAT.TRANSFL will be captured in BB since it was implicitly included in $*.*.* and excluded nowhere for this logger.

Example 6-2 Sample Logger Parameter File

```
-- Logger configuration for two Loggers
LOG $D3.GGSLOG.AA, PROCESS $GGL00, MEGABYTES 500, NUMFILES 10, SECURE "NUUU"
CPU 9,4
FILE $DATA4.*.*
FILE $DATA5.*.*
EXCLUDEFILE $DATA4.REPORTS.*
EXCLUDEFILE $DATA4.DAT.TRANSFL
LOG $D15.GGSLOG.BB, PROCESS $GGL01, MEGABYTES 100, NUMFILES 10, SECURE "NUUU"
CPU 3,2
FILE $*.*.*
EXCLUDEFILE $DATA4.REPORTS.*
```

Logger GGL00:

Configuring Logger and GGSLIB

Run the ADD LOGGER command to process the configuration in LOGPARM. This step establishes a configuration for both Logger and GGSLIB and pre-allocates disk files for each Logger process to use for logging database updates.

Before starting Logger, Oracle GoldenGate must process and store its configuration. This step pre-allocates file space for each log trail to ensure extracted records can be stored.

To process the Logger configuration, enter the following command.

```
GGSCI> ADD LOGGER
```

Starting Logger

To start Logger:

1. Start GGSCI.
2. Enter START LOGGER.

   ```
   GGSCI> START LOGGER
   ```

   By default, this command starts the logger group $GGLnn. If, for example, you have three LOG entries in the LOGPARM file, START LOGGER starts three processes, named $GGL00, $GGL01 and $GGL02.

Using Macros to Bind GGSLIB to a Non-TMF Application

Use the GGSCI BIND PROGRAMS command to bind the GGSLIB library to your non-TMF application. This step also binds GGSLIB to existing user libraries the application calls.

```
TACL > RUN GGSCI
GGSCI> BIND PROGRAMS
```

BIND PROGRAMS prompts for a list of programs to bind with GGSLIB. GGSCI analyzes this list to see which files are already bound and which ones it must bind.

In this context, bound means that GGSLIB runs as a user library in the application program (`BIND CHANGE LIBRARY GGSLIB` in *program_name* is performed). GGSLIB is not literally bound with the application program. If a program already calls a user library, that library is literally bound with GGSLIB to create a new library. The library will have the same name as the old user library.

Building GGSLIB

GGSLIB, built as part of installation, contains the BASELIB module that intercepts Guardian function calls made by the application. GGSLIB also contains C, CRE and COBOL run-time libraries that call Guardian functions. When bound to GGSLIB, these libraries attempt to call the operating system function, but actually call the Oracle GoldenGate function instead. GGSLIB in turn calls the intended operating system function transparently to the application. GGSLIB uses a shared extended memory segment for efficient configuration storage, and maintains a private memory segment for working storage variables.

Without the presence of these libraries, the C and COBOL run-time libraries would be called at the operating system level and would bypass Oracle GoldenGate intercept functions.

Therefore, build these libraries carefully. Keep the following libraries up-to-date with your latest operating system release and related IPMs. Not all of these libraries are required in the GGSLIB build if your application does not run COBOL74, COBOL85 or C routines. It is recommended, however, to bind each of these components that exist on your system into GGSLIB.

Library	Function
`$SYSTEM.ZCOBOLRT.CLIBOBJ`	COBOL74 routines
`$SYSTEM.ZCOB85RT.C8LIB`	COBOL85 routines
`$SYSTEM.ZCRERTL.CFELIB`	Common Run-time Environment
`$SYSTEM.SYSTEM.CRELIB`	Common Run-time Environment
`$SYSTEM.SYSTEM.COBOLLIB`	More COBOL85 routines

To build a new version of GGSLIB, issue the following command from TACL.

```
> RUN BUILDMAC
```

In some Guardian releases there are insignificant conflicts between functions that appear in more than one of the above libraries. You can safely ignore the resultant `BIND` warnings during the build.

Private Memory and Stack Space

GGSLIB routines minimize stack space requirements. By doing so, programs are ensured that typical activities will have enough stack room left for themselves.

For its own working space, GGSLIB allocates a small private memory segment to handle in-transit I/O buffers and keep its own state variables. This requires approximately 250 words.

Alternate Methods of Binding GGSLIB to an Application

There are alternatives to using the Oracle GoldenGate macros (`NLDLIB` for example) to bind the Oracle GoldenGate intercept library to your application. These alternatives may vary depending on your NonStop environment.

For non-native mode systems, a type 100 object file is produced using the TAL, COBOL, or C language compilers. Native mode Itanium systems use EPTAL, ECOBOL or CCOMP to compile type 800 objects.

Using the ?Search Directive

You can connect the application to the Oracle GoldenGate intercept library by using the `?SEARCH` directive in the compile. This copies the library into the application object file. The drawback to this method is that an upgrade to the Oracle GoldenGate application or the operating system will not be picked up by the built-in modules of these programs. A recompile is required to replace the modules.

Non-Native Environments

You can bind the intercept library to application programs in non-native environments by using:

- The NonStop `BIND` utility

  ```
  BIND CHANGE LIBRARY $vol.subvol.library IN application_object
  ```

- A `/LIB /` parameter in the run statement

  ```
  RUN application_object/LIB $vol.subvol.library/
  ```

- `SET SERVER GUARDIAN-LIB` parameter if it is a Pathway server

Native Mode Itanium Systems

The native mode Itanium system does not require any special steps. The intercept library can be bound to the application by any of the following.

- Using the TNS/E Link edit (ELD) utility change command

  ```
  ELD -CHANGE LIBNAME '$vol.svol.library' application_object
  ```

- A `/LIB /` parameter in the run statement

  ```
  RUN application_program/LIB Oracle_GoldenGate_library/
  ```

- Using the server parameter `GUARDIAN-LIB.`

Libraries for Native Applications

If your NonStop environment is running in native mode, you may decide to use native mode libraries so processes run more efficiently. You must use native mode modules and libraries if you are using the encryption or compression capabilities of Oracle GoldenGate. Oracle GoldenGate provides a TACL macro, `NLDLIB`, for building the following native libraries:

- **BASELIBR**: A relinkable, native version of `BASELIB`, a module that intercepts function calls made by the application.
- **GGSDLL**: A native version of `BASELIB` for use as a dynamically linked library (`DLL`) on the operating systems.

- **GGSLIBR**: A relinkable, native BASELIB containing CRE and COBOL SRLs.

> **Note**
>
> Applications running on the operating systems that include native C, native COBOL, and pTAL require two intercept libraries. The one to be linked to the C and COBOL applications should reference the COBOL and CRE dynamic -link libraries, and the one for pTAL should not. This is due to the limitation that pTAL does not perform the necessary initialization of the run-time environment.

Running NLDLIB

Running the NLDLIB macro lets you create these libraries and combine them with the native mode Oracle GoldenGate BASELIBN library and certain Guardian system libraries. You can run NLDLIB as part of your initial installation routine or on its own.

1. Run the following:

   ```
   TACL> RUN NLDLIB
   ```

2. The NLDLIB macro runs, and you are asked if you want to include a user library. Reply either **Y** or **N**.

   ```
   NLDLIB builds the native relinkable object GGSLIBRA Native User Library will be
   built as GGSDLLEnter X at any prompt to EXIT
   Do you want to include your own User Library (Y/N): N
   ```

3. You are asked if you want combine your own library.

4. You are asked if you want to change the location of the AUDCFG.

   ```
   Do you want to change the location for the AUDCFG segment(Y/N) : N
   ```

   If you respond with yes, it prompts you for the new default location (*$vol.subvol*) of the AUDCFG segment.

   NLDLIB builds the GGSLIBR and GGSDLL libraries, displaying a series of informational messages and the names of the files that were created.

5. When the libraries are built, add the new relinkable library to your program object using the LINK PROGRAMS command in GGSCI. You could also run the program as follows:

   ```
   TACL> RUN your_program_name /LIB new_library_name/
   ```

You can also run the NLDLIB macro from the TACL prompt providing arguments in the command line. This is not recommended, however, as it may produce unexpected results. Interactive responses help ensure the appropriate options for your environment.

Removing a Library

To remove the Oracle GoldenGate library from your application, run your program with an empty LIB parameter as follows:

```
TACL> RUN your_program_name /LIB/
```

Activating Authorization of Bound Libraries

You can add the Oracle GoldenGate SFGEXIT module to Safeguard to produce a warning for any program that opens non-audited files for update and does not have the Oracle GoldenGate intercept library bound to it. See "Authentication for Bound Programs" for more information.

> **Note**
>
> Opens on SQL tables, unstructured files, and TMF protected files are always ignored. Opens from processes on remote nodes are also ignored.

You can enable the authorization event and supply optional PARAM-TEXT arguments when the program is added.

The syntax for the ADD within Safecom is:

```
=ADD EVENT-EXIT-PROCESS OPENCHECK PROG $vol.subvol.SFGEXIT
[, PNAME process_name]
[, ENABLE-AUTHORIZATION-EVENT {ON | OFF}]
[, ENABLE {ON | OFF}]
[, PARAM-TEXT {
[, DETAIL]
      [, OSOPENSUMMARY | OSOPENDETAIL | NOOSOPENS]
      , AUDCFG filename [REJECT]

              }
]
```

Option	Description
ENABLE-AUTHORIZATION-EVENT ON | OFF	You can set the authorization event to ON during the ADD of the event. If it is not set, ENABLE-AUTHORIZATION-EVENT will default to OFF. You can set it to ON after the ADD by using an ALTER command.
ENABLE ON | OFF	You can also set ENABLE to ON if ENABLE-AUTHORIZATION-EVENT is set to ON during the ADD. If not set, ENABLE will default to OFF. You can set ENABLE to ON using the ALTER command.
PNAME *process_name*	You can optionally enter a logical process name.

Option	Description
PARAM-TEXT	PARAM-TEXT has the following options: DETAIL DETAIL specifies that a message should be logged to EMS every time a user application that is not bound to the Oracle GoldenGate library opens a file for update. The default is to display a message only the first time the application opens a file for update. **Note:** Use the DETAIL option with care. It may produce a large number of EMS messages due to OPENS for alternate key files and partitions. OSOPENDETAIL | OSOPENSUMMARY | NOOSOPENS OSOPENDETAIL and OSOPENSUMMARY both specify that messages will be logged for OS processes (programs started from $SYSTEM.SYSTEM and $SYSTEM.SYSnn). NOOSOPENS will not log warnings for this type of process. The default is NOOSOPENS. OSOPENDETAIL further specifies that messages should be logged to EMS every time a process that is not bound to the Oracle GoldenGate library opens a file for update. OSOPENSUMMARY displays a message only the first time the process opens a file for update.
AUDCFG *filename* [REJECT]	Identifies the Logger audit configuration file as *filename*. When this option is used, only files matching an entry in the indicated AUDCFG file are acted upon. If a file not in the AUDCFG is opened for update, SFGEXIT replies NO RECORD without applying any processing. If REJECT is specified, an open will be refused when a program does not have GGSLIB bound in and it tries to open a file listed in AUDCFG. **Note:** Using wildcards in the file list may generate an unexpected number 48 errors if it causes the tracking of files that the application would not typically open.

Managing the Authorization Event

Perform the following steps to manage the authorization event:

Adding and Verifying the Authorization Event

The following steps show examples that add, set options, check the status, and remove the authorization event.

1. To add the authorization program, access SAFECOM and enter the ADD statement as shown in the following example.

```
>SAFECOM
=ADD EVENT-EXIT-PROCESS OPENCHECK PROG $DATA1.GGS.SFGEXIT, PNAME $ZSEEP, ENABLE-
AUTHORIZATION-EVENT ON, ENABLE ON, PARAM-TEXT DETAIL
```

2. The INFO command can be used to verify the addition, check the location of the program, check the status of the event, and review the PARAM-TEXT options:

```
=INFO EVENT-EXIT-PROCESS OPENCHECK
EVENT-EXIT-PROCESSOPENCHECK
ENABLED = ON
RESPONSE-TIMEOUT =5 SECONDS
ENABLE-AUTHORIZATION-EVENT= ON
ENABLE-PASSWORD-EVENT = OFF
```

```
PROG= $DATA1.GGS.SFGEXIT
LIB = * NONE *
PNAME = $ZSEEP
SWAP= * NONE *
CPU =2
PRI =155
PARAM-TEXT = DETAIL
```

3. To remove the `OPENCHECK` event, you must first turn off the activation with the `ENABLE OFF`.

```
=ALTER EVENT-EXIT-PROCESS OPENCHECK, ENABLE OFF
=DELETE EVENT-EXIT-PROCESS OPENCHECK
```

4. To exit SAFECOM:

```
=EXIT
```

Using Different PARAM-TEXT Options

Other examples of setting options when adding the authorization event are shown below.

1. The following adds an `OPENCHECK` event that will issue a warning each time the application opens a file that does not have the intercept library bound. It also issue warnings for each open for programs that are run from `$SYSTEM.SYSTEM and $SYSTEM.SYSnn`.

```
=ADD EVENT-EXIT-PROCESS OPENCHECK PROG $DATA1.GGS.SFGEXIT, PARAM-TEXT DETAIL,
OSOPENSDETAIL
```

2. The following adds an `OPENCHECK` event that will issue only one warning for each file without an intercept library, evaluate only files listed in the audit configuration file `$DATA1.GGS.AUDCFG`, and not include programs that are run from `$SYSTEM.SYSTEM` and `$SYSTEM.SYSnn.`

```
=ADD EVENT-EXIT-PROCESS OPENCHECK PROG $DATA1.GGS.SFGEXIT, PARAM-TEXT
AUDCFG $DATA1.GGS.AUDCFG
```

Getting the Current Status of the Authorization Event

Additional examples of monitoring the process are shown below.

1. The following `SEND` *process* `GETSTATS` command retrieves statistics from a running authorization event.

```
GGSCI (\NY) 2445> SEND $ZSEEP, GETSTATS

\NY.$ZSEEP Stats at 2013-08-15 15:14:01.770337
Started 2013-08-15 15:07:47.444913 CPUTime 0:00:00.007707  (PerOp 31)
Audcfg \NY.$data01.zlogdat.audc  fgModtime 2013-08-05 12:10:00.845007
PoolGets            7    PoolPuts         0
GGSRequests        10    Other            0
  SFGRequests     205    Total          245
  Access          205    RemoteNode       0
  NonDisk          50    SQL              0
  Open            154    Readonly       129
  Audcfg Check      2    Found            2    Excluded      0
  Diskfiles       154    Unstruct         0
  TMF Audited       0    SQL Tables       0
  ProcessInfo       0    Cached           0    Errors        0
  FileInfo          2    Cached           0    Errors        2
  GGSProgs          0    SystemProgs      0
  Reported          0    OpensDenied      0
Hash Stats
  Buckets        7919
```

```
  Entries           2    Lookups           2
  Collisions        2    Depth             0
```

2. The following SEND *process* PROCESSINFO command retrieves information on the process.

```
GGSCI (\NY)> SEND $zseep, PROCESSINFO 3,1192

3,1192 \NY.$QA01.BV95014.REPLICAT GGS Code Has Lib
```

3. The following example errors were retrieved by the SEND *process* GETERRORLIST command.

```
2013-08-15 15:12:49.911382
 FILE_GETINFOLISTBYNAME_ error 11 on \NY.$SYSTEM.SYS07.INSPLOG
2013-08-15 15:13:52.254180
 FILE_GETINFOLISTBYNAME_ error 11 on \NY.$DATA01.QA.TESTFILE
```

Working with Parameter Files

Parameters give you complete control over all aspects of Oracle GoldenGate, such as:

- Data selection, mapping, and transformation
- Replication
- Error resolution
- Logging
- Status and error reporting
- System resource usage
- Startup and run-time activities

There can be only one active parameter file for each Manager, Extract, or Replicat. There are two types of parameters: global and file-specific.

- Global parameters apply to all tables specified in the parameter file for synchronization. Some global parameters affect processing while others affect such things as memory utilization.
- File or table-specific parameters control processing for tables specified with a FILE, TABLE or MAP statement. Table-specific parameters enable you to designate one set of processing rules for some tables, while designating other rules for other tables. There are two implementations for file-specific parameters:
  - Toggling the parameter on and off around one or more FILE, TABLE or MAP statements.
  - Adding the parameter within MAP statement so it applies only to that table or file.

Some parameters, such as HANDLECOLLISIONS/NOHANDLECOLLISIONS can be included in a MAP statement or toggled ON and OFF. Others can be implemented using only one of the methods. For further details, see Oracle GoldenGate Parameters.

The ordering of parameters in a parameter file can be important.

- A global parameter can appear anywhere in the parameter file, and it should only be listed in the file once. When listed more than once, only the *last* instance of the parameter is active. All other instances are ignored.
- Table-specific parameters take effect in the order that each parameter is listed in the file.

Table 6-1 Basic Extract and Replicat Parameter Files

Sample Extract parameter file	Sample Replicat parameter file

`EXTRACT NYTOLA`
`DISCARDFILE =DISCARD_FILE, PURGE`
`EXTTRAIL $DATA1.EXTDAT.XX`
`FILE $DATA2.FINANCE.ACCOUNTS;`
	`REPLICAT NYTOLA`
	`DISCARDFILE =$DATA.GGSDISC.NYTOLA, PURGE`
	`ASSUMETARGETDEFS`
	`MAP $DATA2.FINANCE.ACCOUNT,`
	`TARGET $BACK.FINANCE.ACCOUNTS;`

Creating a Parameter File

From the subvolume where Oracle GoldenGate is installed, create a parameter file using the NonStop text editor. The name of the parameter file is usually the same as the process group name. For example, if you created the Extract group `ADD EXTRACT NYTOLA`, you would create your parameter file by entering `TEDIT PARAMS NYTOLA`.

To create a parameter file through GGSCI

1. From the Oracle GoldenGate installation location, run the GGSCI command-line user interface.
2. In GGSCI, issue the following command to open the default text editor.

   ```
   GGSCI> EDIT PARAMS group_name
   ```

   Where:

   group_name is either `MGRPARM` (for the Manager process), `LOGPARM`, or the name of the Extract or Replicat group for which the file is being created. The name of an Extract or Replicat parameter file must match that of its process group.

   Examples:

   - The following creates or edits the parameter file for an Extract group named `EXTORA`.

     ```
     GGSCI> EDIT PARAMS EXTORA
     ```

   - The following creates or edits the parameter file for the Manager process.

     ```
     GGSCI> EDIT PARAMS MGRPARM
     ```

   - The following creates or edits the parameter file for the Manager process.

     ```
     GGSCI> EDIT PARAMS LOGPARM
     ```

3. Using the editing functions of the editor, enter as many comment lines as you want to describe this file, making certain that each line is commented out by two hyphens (--). As an alternative, you can use the `COMMENT` parameter, which causes everything on the same line as the `COMMENT` parameter to be ignored. The syntax for `COMMENT` is:

   ```
   COMMENT comment_text
   ```

   > **Note**
   >
   > Do not put a dash or pound symbol before the `COMMENT` keyword. Do not use `COMMENT` if any column names in the tables contain the word “comment." Instead, use double hyphens (--).

4. On non-commented lines, enter the parameters for your synchronization configuration, starting a new line for each parameter statement.

   For parameters that accept table names, you can use an asterisk (*) wildcard to match any number of characters.

   Parameters have the following general syntax:

   ```
   parameter argument [, option] [&]
   ```

   Where:

   - *parameter* is the parameter name.
   - *argument* is a required argument for the parameter. Some parameters take arguments, while others do not. Separate all arguments with commas, as in the following example:

     ```
     USERID ggs, PASSWORD ggs123
     RMTHOST sysb, MGRPORT 8040
     RMTFILE $DATA05.GGSDAT.C1, PURGE
     ```

     > **Note**
     >
     > Use a maximum of six characters to name any volume that identifies files or tables in parameter files. `$DATA05` is supported, but `$DATA011` is not.

   - *option* is an optional argument.
   - `&` enables you to continue a parameter's arguments on another line. Place it at the end of the line to be continued.

     > **Note**
     >
     > Ampersands (`&`) are not always required to span more than one line, but it is a good practice to use ampersands when:
     >
     > – A parameter is not terminated by a semicolon and the line extends beyond 79 characters
     >
     > – A line for any of the options used for the parameter extend beyond 79 characters.

   - Save and close the file.

Storing Parameter Files

By default, parameter files are stored in the `GGSPARM` subvolume. If you are not going to use the default location, create the new location before starting Oracle GoldenGate. You can change this default location using an `ADD DEFINE` parameter in the `GLOBALS` parameter file, such as the one in the following example.

```
TACL> ADD DEFINE =GGS_PARAMS, CLASS DEFAULTS, VOLUME $VOL.SUBVOL
```

Once paired with a process, a parameter file must remain in its original location for Oracle GoldenGate to operate properly.

Viewing a Parameter File

You can view a parameter file by issuing the GGSCI `VIEW PARAMS` command.

```
VIEW PARAMS filename
```

`VIEW PARAMS` displays the file.

Table 6-2 summarizes the various ways in which you can scroll through its contents.

Table 6-2 Parameter File Scrolling Commands

Command	Description
return, `n`	Next page
/string	Search for next occurrence of *string* in file
number	Go to line indicated by *number*
`l`	Go to last page of file
`b`	Go backwards one page in file
`q`	Quit display
`h`	Help

Reference for Oracle GoldenGate on HP NonStop Guardian has a complete list of commands.

Changing a Parameter File

You make changes to an Oracle GoldenGate NonStop process parameter file by editing it using the NonStop text editor or some other compatible editor.

To ensure that all changes you make to the Manager parameter file are activated you must stop and restart the Manager process. To change an Extract or Replicat parameter file, make your changes, then verify them with the `CHECKPARAMS` parameter as described in "Verifying a Parameter File".

Using OBEY and Macros in Parameters

You can leverage existing parameter files through the Oracle GoldenGate macros and the `OBEY` command. To simplify the process, you can use Oracle GoldenGate macros for a variety of operations, including implementing multiple uses of a statement, consolidating multiple commands, or invoking other macros. You also can use `OBEY` to direct Oracle GoldenGate to retrieve parameter settings from another parameter file. Upon encountering `OBEY`, Oracle GoldenGate processes the parameters from the other file and then returns to the current file to process any remaining instructions.

See "Configuring Custom Operations" for more information about using macros and `OBEY` files.

Verifying a Parameter File

Use the following procedure to confirm that the syntax in an Extract or Replicat parameter file is correct:

1. Include `CHECKPARAMS` in the parameter file.
2. Start the associated process.
3. Oracle GoldenGate audits the syntax and writes the results to the report file or screen. View the report by issuing the following:

   ```
   GGSCI> INFO process_type group_name
   ```

   For example:

   ```
   GGSCI> INFO REPLICAT REPCUST
   ```

   - If the syntax is correct, remove the `CHECKPARAMS` parameter and start the process again to begin processing.
   - If the syntax is wrong, edit the file to correct the syntax based on the report's findings, and then start the process again.

Substituting a Parameter

It is possible to assign different values to a parameter within a parameter file. One-off change synchronization runs that require specific parameters can process with the same parameter file as your default change synchronization routine; any difference in parameter requirements is handled by parameter substitution. This minimizes your need for multiple parameter files.

To include a run-time parameter within the parameter file, precede any intended parameter name with a question mark. Then, before running the Extract process, use the `TACL PARAMS` command to pass the value.

When you are ready to run your special data run, specify the following from your TACL prompt:

```
TACL> PARAM EXTFILE $DATA2.GGS.EXTFILE
TACL> PARAM TABNAME $DATA3.MYDB.ACCOUNTS
TACL> PARAM REGION EAST
TACL> RUN EXTRACT /IN PARMFL/
```

Extract will interpret the parameter as follows:

```
SOURCEISFILE
EXTFILE $DATA2.GGS.EXTFILE
TABLE $DATA3.MYDB.ACCOUNTS, WHERE (REGION = "EAST");
```

> **Note**
>
> A question mark can also be used as a wildcard so care should be exercised in using `PARAMS` and wildcards together. The program will process parameter substitutions first, before evaluating wildcards. It cannot distinguish, however, between `?DATA` as a parameter and `?DATA` as a wildcard, so it is important that the user selects parameter names that are never used as part of an actual file name.

Example 6-3 Parameter File Contents

```
SOURCEISFILE
EXTFILE ?extfile
TABLE ?tabname, WHERE (REGION = "?region");
```

7

Configuring Custom Operations

You can use the custom operations to manipulate Oracle GoldenGate to your specific organization requirements.

You can write C or COBOL routines and call them with Oracle GoldenGate user exits. You can also save frequently used Oracle GoldenGate routines as macros then call the macros from within Extract or Replicat parameter files. You can use `OBEY` files to access frequently used Oracle GoldenGate parameters.

This topic includes the following:

User Exits

User exits allow you to extend and customize the functionality of Extract and Replicat. At different points during Extract and Replicat processing, you can call COBOL, C or TAL routines to perform an unlimited number of functions. You can also easily add functions to the application and respond to database events almost as soon as they occur without altering production programs. For example, user exits can:

- Perform arithmetic operations, special date conversions or table lookups while mapping from one file format to another.
- Implement record archival functions off-line.
- Respond to unusual database events in custom ways, for example, by sending a formatted e-mail message or paging a supervisor based on some field value.
- Accumulate totals and gather statistics.
- Clean up invalid data.
- Determine the net difference in a record before and after an update.
- Accept or reject records based on complex criteria.
- Normalize a database during conversion.
- Eliminate indexes that exist to identify recently changed records.

Record Formats for User Exits

User exits expect records to have a specific format. For example, user exits expect:

- Deletes, inserts, and updates to appear in the buffer as full record images
- Non-compressed data to have no offset or length preceding data
- Compressed Enscribe and SQL updates to both have the following format:

  ```
  (offset)(length)(value)(offset)(length)(value)(. . .)
  ```

where

- `(offset)` is the offset into the Enscribe record of the data fragment that changed.
- (length) is the length of the fragment.

- `(value)` is the data. Fragments can span field boundaries, so full fields are not always retrieved (unless compression is off or `FETCHCOMPS` is used).
- Enscribe has an I/O type of 11; SQL has an I/O type of 15. All other I/O types for deletes, inserts, and updates are in non-compressed format.

> **Note**
>
> The above record formats only apply to data sourced from an HP NonStop system.

Creating User Exits

Create user by performing the following process.

To implement user exits:

1. Create a user exit shell routine in C, TAL or COBOL. The user shell routine is the communication point between Extract or Replicat and your routines.
   - **C shell routines.** Shell routines written in C must be named `CUSEREXIT` and must accept the `EXIT-CALL-TYPE`, `EXIT-CALL-RESULT`, `EXIT-PARAMS`, and `EXIT-REC-BUF` parameters. These parameters are supplied by Oracle GoldenGate in the `XLIBC` include file.
   - **COBOL shell routines.** Shell routines written in COBOL must specify the `ENV COMMON` directive, and the `PROGRAM-ID` of one of the modules must be named `COBOLUSEREXIT`. The `COBOLUSEREXIT` program must have a linkage section that contains `EXIT-CALL-TYPE`, `EXIT-CALL-RESULT`, `EXIT-PARAMS`, and `EXIT-REC-BUF` parameters. These parameters are supplied by Oracle GoldenGate in the `XLIBCOB` copy library.
   - **TAL shell routines.** Shell routines written in TAL must be named `TALUSEREXIT` and must accept the `EXIT-CALL-TYPE`, `EXIT-CALL-RESULT`, `EXIT-PARAMS`, and `EXIT-REC-BUF` parameters. These parameters are supplied by Oracle GoldenGate in the `XLIBTAL` include file.

   See COBOLUSEREXIT for details about COBOL and C programming language routines.
2. Include Calling Environment Functions to retrieve information such as record buffers and transaction contexts, if necessary. If the user exit is written in C, you must include the `USRDECS` file. If the exit is written in COBOL, you must furnish a `CONSULT` directive to either Extract or Replicat. If the exit is written in TAL, you must source the `USRDECT` file.
3. In any language, create routines to respond to each type of event generated by Extract and Replicat.
4. Compile and bind the shell routine and the routines that respond to individual events, creating the user exit module.
5. Link the user exit module with Extract or Replicat by running the `NLDEXIT` macro and creating a custom Extract or Replicat module with a different name.
6. Include the `CUSEREXIT, COBOLUSEREXIT` or `TALUSEREXIT` parameter in your Extract or Replicat parameter file.
7. Run the custom Extract or Replicat module.

Linking User Exits

Run NLDEXIT to link your user exit to the Extract or Replicat.

```
TACL >RUN $vol.subvol.NLDEXIT
[options] [object_type]
```

Options	Description
USEROBJ	The name of the user exit object file.
NEWOBJ	The name of the new object file that will contain the exit routines and the Extract or Replicat module. The file must not yet exist, and will be created in the same subvolume as the Extract/Replicat module.
GGSUBVOL	The fully defined location of the Oracle GoldenGate environment.
CATALOG	The SQL Catalog for the SQLCOMP of the program. This information is not required if your database is Enscribe.
C++VERSION	If there were any C++ modules used, the version of the C++ compiler; 2 or 3.
CEXITWITHCOBOL	Y or N to indicate whether any COBOL modules were used.
SHOWCMD	Displays additional information on the NLDEXIT commands during the session.
HELP	Display NLDEXIT Help text.

Example 7-1 Some Additional Information Displayed with NLDEXIT and SHOWCMD

```
-o $DATA1.GGSSRC.TESTREP
$DATA2.TSPAK.XSKLCON $DATA2.TEST.REPR
$DATA2.TSSOBJ.USRESQL
-nostdfiles
-allow_duplicate_procs
-set runnamed on
-set highpin on
-set highrequesters on
-set saveabend on
-set libname $DATA1.GGSSRC.PRIVLIB
$system.system.crtlmain
-obey $system.system.libcobey
NLD - NATIVE MODE LINKER - T6017D45. . .
(C)1993 Tandem (C)2004 Hewlett-Packard Development Company, L.P.
NLD's command line was:
     \LA.$system.system.nld -stdin
**** INFORMATIONAL MESSAGE **** [20022]:
     The SRL name or archive name specified as 'zcresrl' in a -l, -lib, or
     -import flag was resolved to the SRL named
     '\LA.$SYSTEM.SYS04.zcresrl'.
.
.
. (11 informational messages omitted from this sample)
NLD reported 0 errors.
NLD reported 0 warnings.
NLD reported 12 informational messages.
```

```
NLD created the following type of object file:
     \TRILL.$DATA1.GGSSRC.TESTREP (ELF, executable)
NLD Timestamp:  15DEC2010 15:07:30
Elapsed Time:   00:00:06
```

The following example creates a new native user exit in Extract

Example 7-2 Creating a New Native User Exit

```
TACL> RUN $vol.subvol.NLDEXIT
Creates a new Native Extract or Replicat object file linked with a USEREXIT module.
Enter X at any prompt to quit.

Enter type of GGS object to create Extract or Replicat:
GGS Object type: extract_name
Enter $Vol.Subvol for Extract relinkable installation_location
Enter location of userexit object: your_native_compiled_C_object
Enter name for new object file: new_native_extract

Does your C User Exit contain C++ modules (Y/N): Y
What version compiler was used for C++ (2/3): number
Does your C User Exit contain Cobol modules (Y/N)? Y
New Extract file $vol.subvol.extractname.filename created with user exits.

SQL Catalog for SQLCOMP (or N to avoid SQL compile): SQL_catalog_subvol
```

Sample User Exits

Two sample user exits are supplied with Oracle GoldenGate: `DEMOXCOB` (written in COBOL) and `DEMOXC` (written in C programming language). You can use these exits as skeletons for your own routines.

`DEMOXCOB` illustrates several applications of user exits. `DEMOXCOB` responds to Extract events and performs several tasks, including:

- Mapping data from Enscribe to SQL formats
- Writing a record to an attention log file under certain conditions
- Rejecting records with invalid codes
- Accumulating and outputting order totals
- Writing archive records when delete records are encountered

`DEMOXC` provides an example of how to write a user exit that responds to Replicat events. `DEMOXC` maps records from a source to a target layout and creates a summary transaction record for each delivered transaction.

Using Oracle GoldenGate Macros

By using Oracle GoldenGate macros in parameter files you can easily configure and reuse parameters, commands, and functions. You can use macros for a variety of operations, including:

- Enabling easier and more efficient building of parameters
- Writing once and using many times
- Consolidating multiple statements
- Eliminating redundant column specifications

- Calling other macros
- Creating Macro libraries to share across parameter files.

Oracle GoldenGate macros work with Extract and Replicat parameter files.

Creating a Macro

Create an Oracle GoldenGate macro with the `MACRO` statement.

```
MACRO #macro_name
PARAMS ([param1] [, param2] [...])
BEGIN
macro_body

END;
```

Argument	Description
`MACRO #macro_name`	Defines an Oracle GoldenGate macro. *macro_name* must begin with the # character, as in *#macro1*. If the # macro character is used elsewhere in the parameter file, such as in a table name, you can change it to something else with the `MACROCHAR` parameter. See "Changing the Macro Character" for more information. Macro names are not case-sensitive.
`PARAMS ([param1] [, param2] [...])`	Optional. Used to describe parameters to the macro. Each parameter used in the macro must be declared in the `PARAMS` statement. See "Creating Macro Parameters" for details about this option.
`BEGIN`	Indicates the beginning of the body of the macro. Must be specified before the macro body.

macro_body	Represents one or more statements to be used as parameter file input. *macro_body* can include simple parameter statements, such as
	`COL1 = COL2`
	or more complex statements that include parameters, such as
	`COL1 = #val2`
	In addition, *macro_body* may include invocations of other macros. For example:
	`#colmap(COL1, #sourcecol)`

`END;`	Ends the macro definition.

Creating Macro Parameters

When you specify the optional `PARAMS` statement in a macro, the macro processor reads through the macro body looking for instances of the parameter names you defined in the `PARAMS` statement. For each occurrence of a parameter name, you must specify a corresponding value, which is substituted for the parameter name during invocation.

For example, to convert a proprietary date format, the following macro defines the `#year`, `#month`, and `#day` parameters.

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

```
BEGIN
    @DATE("YYYY-MM-DD", "CC", @IF(#year < 50, 20, 19),
        "YY", #year, "MM", #month, "DD", #day)
END;
```

Parameter values are substituted within the macro body according to the following rules.

1. Parameter names must begin with the # macro character, such as *#param1*. (You can define a different macro character. See "Changing the Macro Character" for details.) When the macro is run, the invocation must include a parameter value for each parameter named in the `PARAMS` statement. Parameter names are not case-sensitive.

   When the macro processor encounters a parameter with the # prefix that is not in the parameter list, the processor determines whether it is an invocation of another macro. Invocations of other macros also begin with the # character, followed by parentheses enclosing a list of parameter values that are separated by commas.
2. Besides the leading # character, valid parameter characters are alphanumeric and can include the underscore character (_).
3. If a parameter name or macro is encountered within quotation marks, it is treated as text and ignored.

Changing the Macro Character

Anything in the parameter file that begins with the # macro character is assumed to be either a macro or macro parameter. This rule does not apply to text within quotation marks; quoted text is ignored.

If the macro character conflicts with a specification in the parameter file, such as table names that include the # character, you specify a different macro character with the `MACROCHAR` parameter. In the following example, $ is defined as the macro character, rather than #.

```
MACROCHAR $
MACRO $mymac
PARAMS ($p1)
BEGIN
col = $p1
END;
```

The `MACROCHAR` can only be specified once, and must be specified before any macros are defined.

Running the Macro

To run a macro, place the run statement in the parameter file at every place you want the process to occur.

```
[target =] #macro_name ([value1] [, value2] [, . . .])
```

Argument	Description
target =	An optional target to which the results of the macro processing are assigned, such as: `DATECOL1 = #make_date(YR1, MO1, DAY1)`
#macro_name	The name of the macro, such as `#assign_date`.

Argument	Description

`([value1] [, value2] [, . . .])`	The parameter values to be substituted inside the macro, such as `#custdate (#year, #month, #day)`. If the optional PARAMS statement is omitted, the parentheses are still required. See the section on invoking macros without parameters on "Invoking a Macro Without Parameters" for more information.
	Valid parameter values include plain text, quoted text, and invocations of other macros. Some examples of valid parameter values are:
	`my_col_1`
	`"your text here"`
	`#mycalc (col2, 100)`
	`#custdate (#year, #month, #day)`
	`#custdate (#getyyyy (#yy), #month, #day)`

Invoking a Macro Without Parameters

If the macro does not specify parameters, the parameter value list is empty, but the parentheses are still required. For example:

```
#no_params_macro ()
```

Sample Macros

This section shows you sample macros for implementing multiple uses of a statement and invoking another macro.

Implementing Multiple Uses of a Statement

You can use macros to implement multiple uses of a statement, and eliminate the need for entering one statement several times.

The following example illustrates how mapping can be improved with a macro. In this example, a proprietary date format must be converted and the process is used several times. For such a scenario, you could implement a date format conversion in a macro similar to the following:

```
MACRO #make_date
PARAMS (#year, #month, #day)
BEGIN
@DATE("YYYY-MM-DD", "CC", @IF(#year < 50, 20, 19),
"YY", #year, "MM", #month, "DD", #day)
END;
```

To run the macro

1. Place the run statements at the appropriate location, similar to:

   ```
   MAP $DATA.PROD.ACCOUNT, TARGET $DATA.BACK.ACCOUNT,
   COLMAP (
   TARGCOL1 = SOURCECOL1,
   DATECOL1 = #make_date(YR1,MO1,DAY1),
   DATECOL2 = #make_date(YR2,MO2,DAY2)
   );
   ```

2. Upon invocation, the macro expands to:

```
MAP $DATA.PROD.ACCOUNT, TARGET $DATA.BACK.ACCOUNT,
COLMAP(
TARGCOL1 = SOURCECOL1,
DATECOL1 = @DATE("YYYY-MM-DD", "CC", @IF(YR1 < 50, 20, 19),
"YY", YR1, "MM", MO1, "DD", DAY1)
DATECOL2 = @DATE("YYYY-MM-DD", "CC", @IF(YR2 < 50, 20, 19),
"YY", YR2, "MM", MO2, "DD", DAY2)
);
```

Consolidating Multiple Commands

In addition, frequently used sets of commands can be specified in a macro, as in this example of the macro `#option_defaults`.

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

Invoking the macro:

```
#option_defaults ()
IGNOREUPDATES
MAP $DATA.PROD.TCUSTMER, TARGET $DATA.BACK.TCUSTMER;
```

expands to:

```
GETINSERTS
GETUPDATES
GETDELETES
INSERTDELETES
IGNOREUPDATES
MAP $DATA.PROD.TCUSTMER, TARGET $DATA.BACK.TCUSTMER;
```

Invoking the macro:

```
#option_defaults ()
MAP $DATA.PROD.TCUSTORD, TARGET $DATA.BACK.TCUSTORD
```

expands to:

```
GETINSERTS
GETUPDATES
GETDELETES
INSERTDELETES
MAP $DATA.PROD.TCUSTORD, TARGET $DATA.BACK.TCUSTORD;
```

Macro Libraries

You can create libraries of macros to be included in different parameter files.

To create a macro library:

1. Create the macros using a text editor, saving them to a file name with the format `$DATA.GGSMACR.`*filename*, where *filename* is the name of the file.

> **Note**
>
> A macro library file can contain multiple macros.

2. Store your macro library files in `$DATA.GGSMACR`.
3. Specify the `INCLUDE` parameter in your parameter file to include the macro library.

Sample Macro Libraries

These samples show:

1. The macro library `$DATA.GGSMACR.DATELIB` that contains `#make_date` and `#assign_date` macros for date conversions
2. The `$DATA.GGSMACR.MAINLIB` macro library containing the macro with multiple commands
3. A sample parameter file calling a macro library that shows the include statement, and invocation statements for one of the macros from the library

   The Extract parameter file is using the `DATELIB` macro library, and the `#assign_date` macro.

Example 7-3 The $DATA.GGSMACR.DATELIB Macro Library

```
--
-- Date macro library
--
MACRO #make_date
PARAMS (#year, #month, #day)
BEGIN
@DATE("YYYY-MM-DD", "CC", @IF(#year < 50, 20, 19),
"YY", #year, "MM", #month, "DD", #day)
END;
MACRO #assign_date
PARAMS (#target_col, #year, #month, #day)
BEGIN
#target_col = #make_date (#year, #month, #day)
END;
```

Example 7-4 The $DATA.GGSMACR.MAINLIB Macro Library

```
--
-- Main macro library
--
INCLUDE $DATA.GGSMACR.DATELIB
MACRO #option_defaults
BEGIN
GETINSERTS
GETUPDATES
GETDELETES
INSERTDELETES
END;
```

Example 7-5 Sample Extract Parameter File

```
-- Parameter file for EXTRACT EXT1
--
INCLUDE $DATA.GGSMACR.DATELIB
EXTRACT EXT1
...
```

```
MAP $DATA.PROD.ACCOUNT, TARGET $DATA.BACK.ACCOUNT,
COLMAP (
TARGCOL1 = SOURCECOL1,
#assign_date(DATECOL1,YR1,MO1,DAY1),
#assign_date(DATECOL2,YR1,MO1,DAY1)
);
...
```

The parameter file processes the macro as follows:

- The `INCLUDE` statement pointing to `DATELIB` is specified at the beginning of the parameter file.
- The `#assign_date` macro is called when needed.

Suppressing Report File Listing

When including long, standard macro libraries, you may want to suppress listing each macro in the report file. Listing can be turned off and on by placing the `LIST` and `NOLIST` commands anywhere within the parameter file or within the included library.

For example, in the following, `NOLIST` suppresses listing each macro in `HUGELIB`. Specifying `LIST` after the `INCLUDE` statement restores listing to the report file.

```
NOLIST
INCLUDE $DATA.GGSMACR.HUGELIB
LIST
EXTRACT EXT1
.
.
.
```

Tracing Parameter Expansion

You can trace macro expansion with the `CMDTRACE` parameter. When `CMDTRACE` is enabled, the macro processor displays macro expansion steps in the process's report file.

The syntax is:

```
CMDTRACE [ON | OFF | DETAIL]
```

Argument	Description
`ON`	Enables tracing.
`OFF`	Disables tracing. This is the default setting.

Tracing is enabled before `#testmac` is called, then disabled after the macro runs, as shown in the following example:

```
EXTRACT EXT1
MACRO #testmac
BEGIN
COL1 = COL2,
COL3 = COL4
END;
.
.
.
```

```
CMDTRACE ON
MAP $DATA.TEST.TEST1, TARGET $DATA.TEST.TEST2,
COLMAP
(
#testmac
);
CMDTRACE OFF
.
.
.
```

Using OBEY Files

With OBEY files, you can direct Oracle GoldenGate to parameters stored in a different file, then return processing to the current parameter file. OBEY files are useful for frequently used parameter statements, or parameters that are used by multiple parameter files.

```
OBEY filename
```

To use an OBEY file:

1. Use the NonStop editor to create a file and enter the desired parameters.
2. Edit the file where you want to place an OBEY parameter.
3. Enter the OBEY parameter, specifying the name of the file as *filename*.

   ```
   OBEY filename
   ```

   For example:

   ```
   OBEY $DATA03.GGS.FINANCE
   ```

Creating High Pin Processes

Use the PCREATE library to intercept the C run-time creation of new processes to create high pin processes.

> **Note**
>
> The PCREATE intercept is only available for native mode on the operating systems.

Replicat

For Replicat PCREATE must be combined with the relinkable PRIVLIB to build a combined library that will include intercepts to create a high pin TACL.

The following example combines the PCREATE intercept object, PCREATEO, with the relinkable (R) native mode (N) PRIVLIB to create a new user library named PRIVLIBX.

```
eld -ul -o PRIVLIBX PRIVLIBR PCREATEO -set interpose_user_library on
FUP LICENSE PRIVLIBX
eld -change libname $DATA.GGS1000.PRIVLIBX REPLICAT
```

In the last step the new PRIVLIBX is assigned as Replicat's library. The library name must be fully qualified as shown in the example.

Extract and GGSCI

Extract and GGSCI can use a combined library, such as created in the above example, or PCREATE can be linked into a user library, such as the PCREATEL in the example below.

```
eld -ul -o PCREATEL PCREATEO -set interpose_user_library_on
```

TACL DEFINE

Enter the DEFINE for TACL, =GGS_TACL_PROGRAM, in GLOBALS if it is to be the same for all Extract, Replicat, and GGSCI programs for that Oracle GoldenGate instance. If it is more specific, include it in the Extract or Replicat parameter file. Alternatively it can also be added to TACLLOC or to TACLCSTM.

The following example DEFINE assumes you first FUP DUP $SYSTEM.SYS*nn*.TACL to $SYSTEM.SYS*nn*.TACLHP, turn HighPin ON, then add the DEFINE.

```
ADD DEFINE =GGS_TACL_PROGRAM, CLASS MAP, FILE $SYSTEM.SYSnn.TACLHP
```

8

Integrating Data

You can integrate only the data you require by using parameters, clauses, column mapping, and functions.
This topic includes the following sections:

Selecting Records

You can select specific records to extract or replicate using the `FILTER` and `WHERE` clauses of the `TABLE` or `MAP` parameters. `FILTER` is the more powerful tool, letting you filter records on a variety of criteria. You may specify multiple filters in one `FILE`, `TABLE`, or `MAP` statement. However, `WHERE` is a quick, simple way to select a record that matches a single criteria. You may only have one `WHERE` clause per statement.

Selecting Records with FILTER

Use the `FILTER` clause of `FILE`, `TABLE`, or `MAP` to select specific records within a file or table for Extract or Replicat. `FILTER` uses the Oracle GoldenGate field conversion functions to evaluate whether to process a record. For example, the following statement extracts records in which the price multiplied by the amount exceeds `10000`:

```
TABLE $DATA.MASTER.CUSTOMER, FILTER ((PRODUCT_PRICE*PRODUCT_AMOUNT)>10000);
```

In another example, the following extracts records containing a string `JOE`:

```
TABLE $DATA.MASTER.CUSTOMER, FILTER (@STRFIND(NAME, "JOE")>0);
```

Selecting Records with WHERE

Use the `WHERE` clause in `TABLE` or `MAP` to select specific records within a table to be extracted or replicated.

The `WHERE` clause consists of the following elements and must be enclosed in parentheses.

Element	Example
Columns from the row	`PRODUCT_AMT`
Numeric values	`-123, 5500.123`
Literal strings enclosed in quotation marks	`"AUTO", "Ca"`
Column tests	`@NULL, @PRESENT, @ABSENT` (column is null, present or absent in the record)
Comparison operators	`=, <>, >, <, >=, <=`
Conjunctive operators	`AND, OR`

Element	Example
Grouping parentheses	open and close parentheses ( ) for logical grouping

Arithmetic operators and floating point data types are not supported. To perform more complex selection conditions, use FILTER.

Comparing Fields

Ensure that the variable and value you specify in a comparison match appropriately. Compare:

- Characters with literal string
- Numeric fields with numeric values, which can include a sign and decimal point
- SQL datetime types to literal strings, using the format in which the field is retrieved by a program

Compressed Update Considerations

When a compressed update record is encountered for a table, only part of the record image is available for the condition evaluation. By default, when a column required by the condition evaluation is missing, the record is ignored and output to the discard file, and a warning is issued.

- Use only columns that appear in the primary key of the record, since key fields are always present in compressed records.
- Test for a column's presence first, then for the column's value.

To test for a column's presence, use the following syntax:

```
field [= | <>] [@PRESENT | @ABSENT]
```

The following example returns all records when the AMOUNT field is over 10000 and does not cause a record to be discarded when AMOUNT is absent.

```
WHERE (AMOUNT = @PRESENT AND AMOUNT > 10000)
```

Testing for NULL Values

Evaluate SQL columns for NULL values with the @NULL clause.

The following test returns TRUE if the column is NULL, and FALSE for all other cases (including a column missing from the record).

```
WHERE (AMOUNT = @NULL)
```

The following test returns TRUE only if the column is present in the record and not NULL.

```
WHERE (AMOUNT = @PRESENT AND AMOUNT <> @NULL)
```

Column Mapping

Oracle GoldenGate provides the capability to transform data between two dissimilarly structured database tables or files. These features are implemented with the COLMAP clause in the TABLE or MAP parameters described in this chapter.

Mapping Between Different Database Structures

Using Oracle GoldenGate, you can transform data to accommodate differences in source and target database structures.

For example:

- The source is a NonStop Enscribe file (ACCTFL), while the target is a SQL table (ACCTTAB).
- 75 fields exist in ACCTFL, while ACCTTAB contains only nine columns.
- Five columns in ACCTTAB have corresponding field names in the ACCTFL (ADDRESS, CITY, STATE, ZIPCODE, SOCIAL_SECURITY_NO).
- A ten digit phone number field in ACCTFL corresponds to separate area code, prefix, and phone number columns in ACCTTAB.
- A date column in ACCTTAB is computed from year, month and day fields in ACCTFL.

In this scenario, you can design a column map in a Replicat parameter file MAP statement on NonStop. For example:

```
MAP $DATA.MASTER.ACCTFL, DEF ACCOUNT-REC,
TARGET $DATA.MASTER.ACCTTAB,
COLMAP (
   USEDEFAULTS,
   NAME = CUST-NAME,
   TRANSACTION_DATE = @DATE ("YYYY-MM-DD",
                      "YY", TRDATE.YEAR,
                      "MM", TRDATE.MONTH,
                      "DD", TRDATE.DAY),
   AREA_CODE = @STREXT (PHONE, 1, 3),
   PHONE_PREFIX = @STREXT (PHONE, 4, 6),
   PHONE_NUMBER = @STREXT (PHONE, 7, 10)
);
```

This statement is composed of the following elements:

1. The source file (ACCTFL) and corresponding DDL definition for ACCOUNT-REC.
2. The target table name (ACCTTAB). No definition is required for the SQL table since it is retrieved automatically from a catalog.
3. The COLMAP parameter.
4. USEDEFAULTS, which directs Replicat to move all fields in ACCTFL that have matching columns in ACCTTAB into the ACCTTAB table. Data translation between different data types is automatic.
5. An explicit assignment of the CUST-NAME field to the NAME column. This is required because the names are different.
6. A date calculation for TRANSACTION_DATE based on three fields in ACCTFL.
7. Extracting parts of PHONE-NO into AREA_CODE, PHONE_PREFIX and PHONE_NUMBER.

Data Type Conversions

Numeric fields are converted from one type and scale to match the type and scale of the target. If the scale of the source is larger than that of the target, the number is truncated on the right. If the target scale is larger than the source, the number is padded with zeros.

Varchar and character columns can accept other character, varchar, group, and datetime columns, or string literals enclosed in quotation marks. If the target character column is smaller than that of the source, the character column is truncated on the right.

Date-time fields can accept datetime and character columns, as well as string literals. If you attempt to map a character into a datetime column, make sure it conforms to the Oracle GoldenGate external SQL format (`YYYY-MM-DD:HH:MI:SS.FFFFFF`). Required precision varies according to data type and target platform. Datetime columns are truncated on the right as necessary. If the source column is not as long as the target, the column is extended on the right with the values for the current date and time.

Oracle GoldenGate User Tokens

Oracle GoldenGate user tokens let you capture data and values for use in data integration. User tokens are composed of alphanumeric data from your source system, database, transactions, and/or records. They can also transfer values into other user tokens generated by queries, procedures, or other called functions.

> **Note**
>
> The user token area is limited to 2000 bytes of information. Token names, data length, and the data itself are all used to calculate the user token area size.

User tokens are stored in each record's trail header, and retrieved by the appropriate Oracle GoldenGate component.

The following tables outline types of data that appear in user tokens.

Table 8-1 Sample Environmental Data for User Tokens

Environmental Detail	Description
`GROUPNAME`	Extract or Replicat group name.
`HOSTNAME`	Host name running the Extract or Replicat.
`OSUSERNAME`	The user name that started Extract or Replicat.

Table 8-2 Sample Header Details and Their Description

Header Detail	Description
`BEFOREAFTERINDICATOR`	Before/after indicator
`COMMITTIMESTAMP`	Commit timestamp
`LOGPOSITION`	Log position
`LOGRBA`	Log RBA

Table 8-2 (Cont.) Sample Header Details and Their Description

Header Detail	Description
TABLENAME	Table name
OPTYPE	Operation type
RECORDLENGTH	Record length
TRANSACTIONINDICATOR	Transaction indicator

Populating User Tokens in the Trail Header

To populate user tokens in the trail header, you must include a TOKEN clause on the FILE or TABLE parameter in the Extract parameter file. To do so, complete the following procedure:

1. Edit the Extract parameter file.

   ```
   GGSCI> TEDIT PARAMS EXTDEMO
   ```

2. Specify a table name

   ```
   TABLE $DATA.MASTER.PRODUCT,
   ```

3. Enter the desired tokens. The @GETENV function, quotation marks and comma delimiter are required.

   ```
   TOKENS
     (
      TKN-GROUP-NAME   =@GETENV ("GGENVIRONMENT", "GROUPNAME"),
      TKN-HOST-NAME    =@GETENV ("GGENVIRONMENT", "HOSTNAME"),
      TKN-OS-USER      =@GETENV ("GGENVIRONMENT", "OSUSERNAME"),
      TKN-BA           =@GETENV ("GGHEADER", "BEFOREAFTERINDICATOR"),
      TKN-COMMIT-TS    =@GETENV ("GGHEADER", "COMMITTIMESTAMP"),
      TKN-LOG-POSITION =@GETENV ("GGHEADER", "LOGPOSITION"),
      TKN-LOG-RBA      =@GETENV ("GGHEADER", "LOGRBA"),
      TKN-TABLE        =@GETENV ("GGHEADER", "TABLENAME"),
      TKN-OP-TYPE      =@GETENV ("GGHEADER", "OPTYPE"),
      TKN-REC-LEN      =@GETENV ("GGHEADER", "RECORDLENGTH"),
      TKN-TRNS-IND     =@GETENV ("GGHEADER", "TRANSACTION INDICATOR"),
     );
   ```

4. Exit the parameter file.

Retrieving Values

To retrieve values, you must include a MAP parameter and a COLMAP clause in the Replicat parameter file, then use the @TOKEN function to specify the values to retrieve.

```
MAP $DATA.MASTER.PRODUCT, TARGET $DATA.MASTER.PRODUCT_CHANGES,
  COLMAP (USEDEFAULTS,
          SOURCE_GROUP      =@TOKEN ("TKN-GROUP-NAME"),
          SOURCE_HOST       =@TOKEN ("TKN-HOST-NAME"),
          SOURCE_USER       =@TOKEN ("TKN-OS-USER"),
          BEFORE_AFTER_IND  =@TOKEN ("TKN-BA"),
          TIMESTAMP         =@TOKEN ("TKN-COMMIT-TS"),
```

```
SOURCE_TABLE      =@TOKEN ("TKN-TABLE"),
IO_TYPE           =@TOKEN ("TKN-OP-TYPE"));
```

The @TOKEN function requires quotation marks.

Default Mapping

When you specify COLMAP USEDEFAULTS, Extract maps columns in the source table to columns in the target with the same name. At startup, Extract outputs column names that match and will map to each other.

The USEDEFAULTS parameter allows matching columns to be mapped, plus additional columns. This can be useful when the source and target definitions are similar but not identical.

If you set up global column mapping rules with COLMATCH parameters, you can map columns with different names to each other using default mapping. See the Extract and Replicat COLMATCH parameter for more details.

When unspecified or no match is found in a default map, a target field defaults to one of the following:

Column	Value
Numeric	Zero
Character or varchar	Spaces
Datetime	Current date and time
Columns that can take a NULL value	NULL

If the target table contains names corresponding to the transactional columns described above, the special column values are mapped to the target record format.

Mapping Examples

The following is the source Enscribe DDL for the examples in this section.

```
RECORD PRODUCT-REC.
FILE IS PRODDAT KEY-SEQUENCED AUDIT.
05 PROD-KEY.
   10 CODE1 PIC X(2).
   10 CODE2 PIC 9(2).
05 PROD-INDEX1.
   10 PRICE PIC 9(7)V9(2) COMP.
   10 CODE1 PIC X(2).
   10 CODE2 PIC 9(2).
05 PROD-INDEX2.
   10 INVENTORY PIC 9(5).
   10 CODE1 PIC X(2).
   10 CODE2 PIC 9(2).
05 DESC     PIC X(40).
KEY IS PROD-KEY.
END.
```

The following is the target SQL DDL for the examples in this section.

```
Target SQL DDL
CREATE TABLE PRODTAB
(
CODE CHAR(4) NOT NULL
, PRICE NUMERIC (8,2) NOT NULL
```

```
, INVENTORY DECIMAL (6)
, MANAGER CHAR (20) NOT NULL
, DESC VARCHAR (30)
, UPDATE_TIME DATETIME YEAR TO SECOND NOT NULL
, PRIMARY KEY (CODE)
);
```

Legal Column Mapping

Note that one can move a group level (PROD-KEY) to a character field. This is feasible since CODE2 is a DISPLAY field, not a COMP. Also, the user does not have to qualify PRICE, INVENTORY or DESC since they are all unique in the source definition. UPDATE_TIME will default to the time at which EXTRACT processes the record. PRICE may be truncated since it has one more significant digit in the source field than in the target.

```
FILE $DAT11.OLDAPP.PRODFL,
DEF PRODUCT-REC,
TARGET $DATA6.NEWAPP.PRODTAB,
COLMAP
  (CODE      = PROD-KEY,
   PRICE     = PROD-INDEX1.PRICE,
   INVENTORY = INVENTORY,
   MANAGER   = "Jack Smith",
   DESC      = DESC);
```

Dangerous Mapping if AUDITCOMPRESS Used on Source File

Since this mapping takes the primary key value from a non-primary key source, it discards the result whenever a source record is updated without updating the price. In the following example, even if AUDITCOMPRESS is used, updates can be delivered since the primary key is always present.

```
FILE $DAT11.OLDAPP.PRODFL,
DEF PRODUCT-REC,
TARGET $DATA6.NEWAPP.PRODTAB,
COLMAP
  (CODE      = PROD-INDEX1.CD1,
   PRICE     = PROD-INDEX1.PRICE,
   INVENTORY = INVENTORY,
   MANAGER   = "Unknown",
   DESC      = DESC);
```

Using Constants, Taking Default Values.

This mapping sets PRICE to zero and Manager to spaces since they are not null fields, and sets INVENTORY and DESC to NULL since they can take null values.

```
TABLE $DAT11.OLDAPP.PRODFL,
DEF PRODUCT-REC,
TARGET $DATA6.NEWAPP.PRODTAB,
COLMAP
  (CODE = PROD-KEY,
   UPDATE_TIME = "2009-01-01:08:00:00");
```

Field Conversion Functions

Using field conversion functions, you can manipulate numbers, strings and source column or field values into the appropriate format for target columns.

See Field Conversion Functions for more information about column conversion functions.

Function Arguments

Column conversion functions can take one or more of the following parameters.

Parameter	Example
A numeric constant	`123`
A string constant	`"ABCD"`
A column or field from the source table or file	`PHONE-NO.AREA-CODE or COLUMN_3`
An arithmetic expression	`COL2 * 100`
A comparison expression	`COL3 > 100 AND COL4 > 0`
A field conversion function	its own parameters

> **Note**
>
> Argument checking at run-time is not always strict and errors in argument passing are sometimes not detected until records are processed.

Arithmetic Expressions

Arithmetic expressions can be combinations of the following elements.

- Numbers
- Columns that contain numbers
- Functions that return numbers
- Arithmetic operators: + (plus), - (minus), * (multiply), / (divide), \ (remainder)
- Comparison operators: > (greater than), >= (greater than or equal), < (less than), <= (less than or equal), = (equal), <> (not equal)
- Parentheses (for grouping results in the expression)
- Conjunction operators: `AND, OR`

To return the result of an arithmetic expression to a column, use the `COMPUTE` function.

The `COMPUTE` function is not required when an expression is passed as an argument, as in `@STRNUM (AMOUNT1 + AMOUNT2, RIGHT)`.

`@STRNUM (@COMPUTE(AMOUNT1 + AMOUNT2), RIGHT)` would return the same result.

Arithmetic results derived from comparisons are zero (indicating `FALSE`) or non-zero (indicating `TRUE`).

When conjunction operators are involved in an expression, only the necessary part of the expression is evaluated. Once a statement is FALSE, the rest of the expression is ignored. This can be valuable when evaluating fields that may be missing or null.

For example, assume the value of COL1 is 25 and the value of COL2 is 10:

```
@COMPUTE (COL1 > 0 AND COL2 < 3) returns 0
@COMPUTE (COL1 < 0 AND COL2 < 3) returns 0 (and COL2 < 3 is never evaluated)
@COMPUTE ((COL1 + COL2)/5) returns 7
```

See Field Conversion Functions for details about the functions.

Null, Invalid, and Missing Columns and Fields

One problem encountered when calculating column values is that some data may be missing from the expression.

Table 8-3 summarizes the status conditions that the source columns or fields may assume.

Table 8-3 Explanation of Null, Invalid, and Missing Columns and field

Column Status	Description
Missing	Frequently, data is missing in compressed update records. Compressed update records contain only those source columns that changed, plus the key of the source file or table.
Null	A source column may contain a null value, which makes a calculation difficult.
Invalid	The source data is invalid.

When one of these conditions occurs, by default the condition is returned as the result of the function.

For example, if BALANCE is 1000, but AMOUNT is NULL, the following expression returns NULL.

```
NEW_BALANCE = @COMPUTE (BALANCE + AMOUNT)
```

As another example, the AMOUNT field is defined as PIC 9(5)V99 in an Enscribe record definition, but contains spaces. In that case, the above expression returns INVALID, and the record is discarded.

If AMOUNT, but not BALANCE, is present in the update record, the field is not mapped.

Overriding Exceptional Conditions

The IF, COLSTAT and COLTEST functions recognize null, invalid, or missing columns and can compute alternative values.

For example:

```
NEW_BALANCE = @IF (@COLTEST (BALANCE, NULL, INVALID) OR
                   @COLTEST (AMOUNT, NULL, INVALID),
                   @COLSTAT (NULL),
                   BALANCE + AMOUNT)
```

This returns one of the following:

- NULL when BALANCE or AMOUNT is NULL or INVALID
- MISSING when either column is missing

- The sum of the columns

Retrieving Before Values

For update operations, it can be advantageous to retrieve the *before* values of source columns: the values before the update occurred. These values are stored in the trail and can be used in **Filters**, **SQLEXEC**, **EventActions**, and **column mapping**. For example, you can:

- Retrieve the before image of a row as part of a column-mapping specification in an exceptions MAP statement, and map those values to an exceptions table for use in testing or troubleshooting.
- Perform delta calculations. For example, if a table has a Balance column, you can calculate the net result of a particular transaction by subtracting the original balance from the new balance, as in the following example:

```
MAP $VOL.SUBVOL.TABLE, TARGET $VOL.SUBVOL.TABLE,
COLMAP (PK1 = PK1, delta = balance – @GETVAL(BEFORE.balance));
```

To Reference the Before Value

1. Use the `BEFORE` keyword, then a dot (.), then the name of the column for which you want a before value, then wrapping the entire clause in a valid @function such as:

   ```
   @GETVAL(BEFORE.<column_name>)
   ```

2. Use the `GETUPDATEBEFORES` parameter in the Extract parameter file to capture before images from the transaction record, or use it in the Replicat parameter file to use the before image in a column mapping or filter. If the database uses compressed updates, using the BEFORE prefix results in a “column missing” condition and the column map is executed as if the column were not in the record. To ensure that column values are available, see "Compressed Update Considerations.

Detokenizing Base24 (Classic) TLF/PTLF records

Oracle GoldenGate for HP NonStpop has been enhanced to provide the same detokenizing functionality by a simple `@function()` call that was previously only available from a User Exit.

```
@GETTLFTOKEN ("value", length, ["NOWARN"]),
```

Where `"value"` = the Token Id

`Length` = the size of the DDL definition created for that token including the `TKN-HEADER`.

Warnings are by default printed to the report file for those tokens which the length does not properly match.

When `@GETTLFTOKEN()` is triggered for the first time, the entire record buffer is parsed once for all tokens, saving each for immediate look up during subsequent functions calls. When a mapped token is found in the table, having been save during parsing, the data portion of the token is moved to the output buffer for the specified length.

Converting from the T24 User Exit to Oracle GoldenGate Mapping

Your Base24 DDL dictionary you created for T24 is still required, but no changes to the dictionary are necessary.

To change the Extract parameters:

1. Remove `CUSEREXIT`.
2. Change (if present) the file name as follows:

   ```
   FILE-NAME = "        "
   ```

   to either (for source filename):

   ```
   FILE-NAME = @strext (
               @getenv ("GGHEADER", "SOURCEFILENAME"),
               @strlen(@getenv ("GGHEADER", "SOURCEFILENAME")) - 7,
               @strlen(@getenv ("GGHEADER", "SOURCEFILENAME"))
                   ),
   ```

   Or (for target filename)

   ```
   FILE-NAME = @strext (
               @getenv ("GGHEADER", "TARGETFILENAME"),
               @strlen(@getenv ("GGHEADER", "TARGETFILENAME")) - 7,
               @strlen(@getenv ("GGHEADER", "TARGETFILENAME"))
                   ),
   ```

   To understand which to choose, the Source or Target, look at the fourth position of `EXITPARAMS 1 = Source, 0 = Target`.
3. Remove the whole `EXITPARAM` from the Map.
4. Change All token column maps by removing `TKN-EYE-CATCHER`, `TKN-ID`, and `TKN-LGTH`.

   Example of change for token B0:

   Remove each of

   ```
   TKNB0-TKN-EYE-CATCHER = "!",
   TKNB0-TKN-ID = "B0",
   TKNB0-TKN-LGTH = 448,
   ```

   Then add the new format column function as

   ```
   TKNB0 = @GETTLFTOKEN ("B0", 448),
   ```

   > **Note**
   >
   > The Header token colmap details must remain.
   > Example:
   >
   > ```
   > HEADER-TKN.HDR-EYE-CATCHER = "&",
   > HEADER-TKN.HDR-CNT = 27,
   > HEADER-TKN.HDR-LGTH = 2474,
   > ```
5. Update the Extract group in GGSCI:

   ```
   Alter <group name>, program Extract
   ```

For more information see **NOTES** in Detokenizing with Oracle GoldenGate Mapping.

Detokenizing with Oracle GoldenGate Mapping

In the case of having never used the Oracle GoldenGate Base24 Application adaptor T24, follow these steps to set up Extract to detokenized TLF/PTLF records:

1. Create DDL Dictionary(s) with definition entries for each required token and a target record layout containing those new token definitions. If your requirement is for both ATM and POS, then because of duplicate definition names containing differing column details, we recommend creating two dictionaries.

   a. Duplicate the relevant ACI provided Base24 files to create the needed dictionaries. Example for ATM:

   ```
   Volume to $VOL.OGGB24A
   Dup DDLGDEFS, CUSTCNST, DDLBATKN, DDLATTKN, and DDLFTLF
   Edit both CUSTCNST and DDLFTLF, commenting out all =Defines
   ddl /in DDLGDEFS, out $s.#defs/dict(n)  n = non audited
   ddl /in CUSTCNST, out $s.#cnst/dict(n)  n = non audited
   ddl /in DDLBATKN, out $s.#batkn/dict(n)  n = non audited
   ddl /in DDLATTKN, out $s.#atkn/dict(n)  n = non audited
   ddl /in DDLFTLF, out $s.#tlf/dict(n)  n = non audited
   Check spooler results.
   ```

   b. Example for POS:

   ```
   Volume to $VOL.OGGB24P
   Dup DDLGDEFS, CUSTCNST, DDLBATKN, DDLPSTKN, and DDLFPTLF
   Edit both CUSTCNST and DDLFPTLF, commenting out all =Defines
   ddl /in DDLGDEFS, out $s.#defs/dict(n)  n = non audited
   ddl /in CUSTCNST, out $s.#cnst/dict(n)  n = non audited
   ddl /in DDLBATKN, out $s.#batkn/dict(n)  n = non audited
   ddl /in DDLPSTKN, out $s.#pskn/dict(n)  n = non audited
   ddl /in DDLFPTLF, out $s.#ptlf/dict(n)  n = non audited
   ```

   Check spooler results.

   c. Create required token definitions.
   Each Token definition to be created will consist of two parts. The `TKN-HEADER` and the token name.

   This example shows how to create an output record `OGGTLF-PINCNG`, that contains the PIN Change token `PINC-TOKEN` and the Non-Currency Dispense token `AT-FLG1-TKN`.

   ```
   DEFINITION TKN06.
   02 TKN-HEADER TYPE *.
   02 PINC-TKN TYPE *.
   END

   DEFINITION TKN24.
   02 TKN-HEADER TYPE *.
   02 AT-FLG1-TKN TYPE *.
   END
   ```

   If you need the optional `FILE-NAME` as part of the output record, you will need to also add this specific definition.

   ```
   DEFINITION FILE-NAME PIC X(8)
   ```

   The resulting definition structures will look like:

   ```
   ?SECTION TKN06,TANDEM
   * Definition TKN06 created on 06/04/2021 at 13:31
     01 TKN06.
   ```

```
    02 TKN-HEADER.
      03 EYE-CATCHER                PIC X.
      03 USER-FLD1                  PIC X.
      03 TKN-ID                     PIC X(2).
      03 LGTH                       NATIVE-2.
    02 PINC-TKN.
      03 NEW-PIN-FRMT               PIC X.
      03 NEW-PIN-OFST               PIC X(16).
      03 PIN-CNT                    PIC X.
      03 NEW-PIN-SIZE               PIC 9(2).
      03 NEW-PIN-1                  PIC X(16).
      03 NEW-PIN-2                  PIC X(16).

?SECTION TKN24,TANDEM
* Definition TKN24 created on 06/04/2021 at 15:26
  01 TKN24.
    02 TKN-HEADER.
      03 EYE-CATCHER                PIC X.
      03 USER-FLD1                  PIC X.
      03 TKN-ID                     PIC X(2).
      03 LGTH                       NATIVE-2.
    02 AT-FLG1-TKN.
      03 CRD-TYP                    PIC X(2).
      03 AUTH-ONLY                  PIC X.
      03 SVC-IND                    PIC X.
      03 EXP-DAT                    PIC X(4).
      03 CASH-DEP-FLG               PIC X.
      03 CRD-STAT                   PIC X(1).

?SECTION FILE-NAME,TANDEM
* Definition FILE-NAME created on 06/04/2021 at 13:34
  01 FILE-NAME                      PIC X(8).
```

Once all of the token definitions are created, create a new record definition for the target format containing all of your token definitions needed for your particular usage, preceded by definitions `FILE-NAME (Optional)`, `HEAD`, `AUTH`, and `HEADER-TKN`.

```
RECORD OGGTLF-PINCNG.
02 FILE-NAME TYPE *.
02 HEAD TYPE *.
02 AUTH TYPE *.
02 HEADER-TKN TYPE *.
02 TKN06 TYPE *.
02 TKN24 TYPE *.
END
```

The resulting definition structures looks as follows:

```
?SECTION OGGTLF-PINCNG,TANDEM
* Record OGGTLF-PINCNG version 1 updated on 06/04/2021 at 15:27
  01 OGGTLF-PINCNG.
    02 FILE-NAME                    PIC X(8).
    02 HEAD.
      03 DAT-TIM                    NATIVE-8.
      03 REC-TYP                    PIC X(2).
      03 AUTH-PPD                   PIC X(4).
      03 TERM.
        04 LN                       PIC X(4).
        04 FIID                     PIC X(4).
        04 TERM-ID                  PIC X(16).
      03 CRD.
        04 LN                       PIC X(4).
        04 FIID                     PIC X(4).
```

```
    04 PAN.
      05 NUM                      PIC X(19).
    04 MBR-NUM                    PIC 9(3).
  03 BRCH-ID                      PIC X(4).
  03 REGN-ID                      PIC X(4).
02 AUTH.
  03 TYP-CDE                      PIC X(2).
  03 TYP                          PIC 9(4).
  03 RTE-STAT                     PIC 9(2).
  03 ORIGINATOR                   PIC X.
  03 RESPONDER                    PIC X.
  03 ENTRY-TIM                    NATIVE-8.
  03 EXIT-TIM                     NATIVE-8.
  03 RE-ENTRY-TIM                 NATIVE-8.
  03 TRAN-DAT.
    04 YY                         PIC X(2).
    04 MM                         PIC X(2).
    04 DD                         PIC X(2).
  03 TRAN-TIM.
    04 HH                         PIC X(2).
    04 MM                         PIC X(2).
    04 SS                         PIC X(2).
    04 TT                         PIC X(2).
  03 POST-DAT.
    04 YY                         PIC X(2).
    04 MM                         PIC X(2).
    04 DD                         PIC X(2).
  03 ACQ-ICHG-SETL-DAT.
    04 YY                         PIC X(2).
    04 MM                         PIC X(2).
    04 DD                         PIC X(2).
  03 ISS-ICHG-SETL-DAT.
    04 YY                         PIC X(2).
    04 MM                         PIC X(2).
    04 DD                         PIC X(2).
  03 SEQ-NUM                      PIC X(12).
  03 TERM-TYP                     PIC 99.
  03 TIM-OFST                     NATIVE-2.
  03 ACQ-INST-ID-NUM              PIC 9(11).
  03 RCV-INST-ID-NUM              PIC 9(11).
  03 TRAN-CDE                     PIC X(6).
  03 TRAN-CDE-R REDEFINES TRAN-CDE.
    04 T-CDE                      PIC X(2).
    04 T-FROM                     PIC X(2).
    04 T-TO                       PIC X(2).
  03 FROM-ACCT.
    04 ACCT-NUM                   PIC X(19).
  03 USER-FLD1                    PIC X.
  03 TO-ACCT.
    04 ACCT-NUM                   PIC X(19).
  03 MULT-ACCT                    PIC 9.
  03 AMT-1                        NATIVE-8.
  03 AMT-2                        NATIVE-8.
  03 AMT-3                        NATIVE-8.
  03 DEP-BAL-CR                   NATIVE-4.
  03 DEP-TYP                      PIC 9.
  03 RESP-CDE                     PIC X(3).
  03 RESP-CDE-R REDEFINES RESP-CDE.
    04 RESP-BYTE-1                PIC X.
    04 RESP-BYTE-2                PIC X(2).
  03 TERM-NAME-LOC                PIC X(25).
  03 TERM-OWNER-NAME              PIC X(22).
```

```
  03 TERM-CITY                  PIC X(13).
  03 TERM-ST-X                  PIC X(3).
  03 TERM-CNTRY-X               PIC X(2).
  03 ORIG.
    04 OSEQ-NUM                 PIC X(12).
    04 OTRAN-DAT                PIC X(4).
    04 OTRAN-TIM                PIC X(8).
    04 B24-POST-DAT             PIC X(4).
  03 ORIG-CRNCY-CDE              PIC 9(3).
  03 USER-FLD2                   PIC X(30).
  03 MULT-CRNCY REDEFINES USER-FLD2.
    04 AUTH-CRNCY-CDE            PIC 9(3).
    04 AUTH-CONV-RATE            PIC 9(8).
    04 SETL-CRNCY-CDE            PIC 9(3).
    04 SETL-CONV-RATE            PIC 9(8).
    04 CONV-DAT-TIM              NATIVE-8.
  03 RVSL-RSN                    PIC 99.
  03 PIN-OFST                    PIC X(16).
  03 SHRG-GRP                    PIC X.
  03 DEST-ORDER                  PIC X.
  03 AUTH-ID-RESP                PIC X(6).
  03 REFR.
    04 IMP-IND                   PIC X.
    04 AVAIL-IMP                 PIC X
                                 OCCURS 2 TIMES.
    04 LEDG-IMP                  PIC X
                                 OCCURS 2 TIMES.
    04 HLD-AMT-IMP               PIC X
                                 OCCURS 2 TIMES.
    04 CAF-REFR-IND              PIC X.
    04 USER-FLD3                 PIC X.
  03 DEP-SETL-IMP-FLG            PIC X.
  03 ADJ-SETL-IMP-FLG            PIC X.
  03 REFR-IND.
    04 PBF1                      PIC X.
    04 PBF2                      PIC X.
    04 PBF3                      PIC X.
    04 PBF4                      PIC X.
  03 USER-FLD4                   PIC X(16).
  03 FRWD-INST-ID-NUM            PIC 9(11).
  03 CRD-ACCPT-ID-NUM            PIC 9(11).
  03 CRD-ISS-ID-NUM              PIC 9(11).
02 HEADER-TKN.
  03 EYE-CATCHER                 PIC X.
  03 USER-FLD1                   PIC X.
  03 CNT                         NATIVE-2.
  03 LGTH                        NATIVE-2.
02 TKN06.
  03 TKN-HEADER.
    04 EYE-CATCHER               PIC X.
    04 USER-FLD1                 PIC X.
    04 TKN-ID                    PIC X(2).
    04 LGTH                      NATIVE-2.
  03 PINC-TKN.
    04 NEW-PIN-FRMT              PIC X.
    04 NEW-PIN-OFST              PIC X(16).
    04 PIN-CNT                   PIC X.
    04 NEW-PIN-SIZE              PIC 9(2).
    04 NEW-PIN-1                 PIC X(16).
    04 NEW-PIN-2                 PIC X(16).
02 TKN24.
  03 TKN-HEADER.
```

```
          04 EYE-CATCHER              PIC X.
          04 USER-FLD1                PIC X.
          04 TKN-ID                   PIC X(2).
          04 LGTH                     NATIVE-2.
       03 AT-FLG1-TKN.
          04 CRD-TYP                  PIC X(2).
          04 AUTH-ONLY                PIC X.
          04 SVC-IND                  PIC X.
          04 EXP-DAT                  PIC X(4).
          04 CASH-DEP-FLG             PIC X.
          04 CRD-STAT                 PIC X(1).
```

2. Create an Extract Pump.

   Sample parameters to write a trail as a detokenize record of columns.

```
EXPANDDDL EXPANDGROUPARRAYS RESOLVEDUPGROUP
DICTIONARY $VOL.OGGB24A

RMTHOST {host_name | ip_address},  MGRPORT port_number
RMTTRAIL <remote file> [, FORMAT RELEASE major.minor]
FILE $VOL.SUBVOL.TL*, TARGET $VOL.SUBVOL.TLYYMMDD,
DEF TLF,
TARGETDEF OGGTLF-PINCNG,
USETARGETDEFLENGTH,
COLMAP (USEDEFAULTS,
        FILE-NAME =  @strext (
                     @getenv ("GGHEADER", "TARGETFILENAME"),
                     @strlen(@getenv ("GGHEADER", "TARGETFILENAME")) - 7,
                     @strlen(@getenv ("GGHEADER", "TARGETFILENAME"))
                              ),
        HEADER-TKN.HDR-EYE-CATCHER = "&",
        HEADER-TKN.HDR-CNT = 3,
        HEADER-TKN.HDR-LGTH = 80,
        TKN06 = @GETTLFTOKEN ("06", 52),
        TKN24 =  @GETTLFTOKEN ("24",10))
WHERE (REC-TYP = "01" OR REC-TYP = "20" OR REC-TYP = "21");
```

NOTES
Determining HEADER-TKN details

`HEADER-TKN.HDR-EYE-CATCHER` A single character that should be set to an ampersand “&” to identify the beginning of the token `HEADER.HEADER-TKN.HDR-CNT`, Two digits that specify the number of tokens plus one for the `HEADER.HEADER-TKN.HDR-LGTH`, Two digits that store the length of the token header area. This is calculated as: the 6 byte header area, plus each token’s length, plus 6 bytes for the token header. In this exapmple, the calculation would be:

6 + (52 + 6) + (10 + 6) = 80.

Ambiguous References

ACI’s definition names often contain duplicate field names from one definition to another. For example the AUTH portion of PTLF contains the field USER-FLD2. These same field names also exists in several token definitions. Because OGG Extract and Replicat map fields by name, these fields are not only mapped correctly for the AUTH portion of the record, but also mapped from the source AUTH USER-FLD2 to every name matching USER-FLD2 in all tokens. This can be seen to corrupt token data fields. Warnings will occur when an ambiguous reference is detected, like

> **Warning**
>
> The default map caused multiple target cols from a single source:
>
> ```
> Offset Source Column Offset Target Column
>
> 321:TERM-CNTRY-CDE =  329:TERM-CNTRY-CDE
>                       2182:TERM-CNTRY-CDE
>
> 327:USER-FLD2      =  335:USER-FLD2
>                       2250:USER-FLD2
>
> 354:TERM-TYP       =   362:TERM-TYP
>                       2977:TERM-TYP
>
> 416:AMT-1          =  424:AMT-1
>                       2472:AMT-1
>
> 424:AMT-2          =  432:AMT-2
>                       2480:AMT-2
>
> 432:EXP-DAT        =  440:EXP-DAT
>                       1128:EXP-DAT
> ```

For this reason it's necessary to ensure the field names are unique from source to target so only the correctly matching fields are mapped. This is done by using the:

`EXPANDDDL EXPANDGROUPARRAYS RESOLVEDUPGROUP`.

Using `EXPANDDDL` option `INCLUDEREDEFS` is not required for using this functionality. It's only required if the target Replicat must specifically map those redefined columns because of data type or size changes. The use of `INCLUDEREDEFS` in this case is only used to describe the redefined columns in trail metadata.

Troubleshooting Shooting Tokens

The map option of `TRACETLFTOKENS` can be used to see what tokens are found in a given record and which are move to the output record. A sample of the report output generated.

```
Tokens found in source record at seqno 184 rba 0
Token id CI at offset 928 for 70 bytes
Token id 04 at offset 1004 for 20 bytes
Token id B4 at offset 1030 for 20 bytes
Token id B2 at offset 1056 for 80 bytes
Token id B3 at offset 1142 for 44 bytes
Token id C1 at offset 1192 for 16 bytes
Token id B7 at offset 1214 for 86 bytes
Token id B8 at offset 1306 for 34 bytes
Token id B9 at offset 1346 for 60 bytes
Token id C4 at offset 1412 for 12 bytes
Token id CH at offset 1430 for 36 bytes
Token id C0 at offset 1472 for 26 bytes
Token id M2 at offset 1504 for 20 bytes
Token id 28 at offset 1530 for 88 bytes
Token id B5 at offset 1624 for 20 bytes
User Data field for Token id QZ at offset 171 is ON (with embedded length of 134)

Returning token QZ from User Data offset 786 to function result for 134 bytes with
embedded length
```

```
Returning token id 04 from input to function result for 20 bytes
Returning token id 28 from input to function result for 88 bytes
Returning token id B2 from input to function result for 80 bytes
Returning token id B3 from input to function result for 44 bytes
Returning token id B4 from input to function result for 20 bytes
Returning token id B5 from input to function result for 20 bytes
Returning token id C0 from input to function result for 26 bytes
Returning token id C4 from input to function result for 12 bytes
Returning token id CH from input to function result for 36 bytes
Returning token id CI from input to function result for 70 bytes
```

Suppressing Expected Warning Messages

When using `@GETTLFTOKEN()`, the group level name is all that is mapped, this causes Extract to issue warning messages for all of the fields within the group that these column have not been mapped.

The following target columns were not mapped:

```
HDR-USER-FLD1
TKN-EYE-CATCHER
TKN-USER-FLD1
TKN-ID
```

> ⚠ **Caution**
>
> After you are sure that all columns that need explicit maps, you can use the parameter `SUPPRESSMISSINGMAPS` to hide these messages. Use with caution.

Logdump

Logdump can be used to see the breakdown of a record.

For a trail produced by `LOGGER`, after reading a record, use Base24Tokens Detail `ON | OFF`

Logdump 3 > recLogdump 4 >Base24Tokens detail on

```
…
…
 Offset  928 Token CI (x4349) Length 70
 2020 2020 3030 3030 2020 2020 2020 2020 2020 2020 |     0000
 2020 2020 4242 3039 3030 3331 2020 2020 2020 2020 |     BB090031
 2020 2020 2020 204e 4e20 2020 2020 2020 2020 2020 |        NN
 2020 2020 2020 2020 2020                          |
Offset 1004 Token 04 (x3034) Length 20
 2032 3030 3030 3030 3037 3035 5920 2020 2020 5920 |  20000000705Y     Y

Logdump 5 > rec
Logdump 6 > Base24Tokens detail off
Header Token at Offset  922, Length  728, Count   15
Offset  928 Token CI (x4349) Length 70
Offset 1004 Token 04 (x3034) Length 20
```

For a trail produced by Extract after using `@GETTLFTOKEN ()` and metadata is present, use Detail Data to see a column details of the record.

```
Column   170 (x00aa), Len     1 (x0001)  HDR-EYE-CATCHER
 26                                      | &
Column   171 (x00ab), Len     1 (x0001)  HDR-USER-FLD1
 20                                      |
Column   172 (x00ac), Len     2 (x0002)  HDR-CNT
```

```
 001b                                                    | ..
Column   173 (x00ad), Len     2 (x0002)  HDR-LGTH
 09aa                                                    | ..
Column   176 (x00b0), Len     1 (x0001)  TKNQZ-TKN-EYE-CATCHER
 21                                                      | !
Column   177 (x00b1), Len     1 (x0001)  TKNQZ-TKN-USER-FLD1
 20                                                      |
Column   178 (x00b2), Len     2 (x0002)  TKNQZ-TKN-ID
 515a                                                    | QZ
Column   179 (x00b3), Len     2 (x0002)  TKNQZ-TKN-LGTH
 00ca
```

The following table list all the standard BASE24 tokens for the BASE, ATM, and POS products.

Table 8-4 BASE - DDLBATKN

TOKEN NAME	ID	NAME	DESCRIPTION
ACCT-QUAL-TKN	18	TKN18	Account Qualifier Token
ACQ-RTE-TKN	BA	TKNBA	Acquirer Routing Token
CR-LINE-TKN	13	TKN13	Credit Line Token
CRD-POSTAL-CDE-TKN	27	TKN27	Cardholder Postal Code Token
DATA-ENCRYPTION-KEY-TKN	BN	TKNBN	Data Encryption Key Token
EMV-DISCR-TKN	B3	TKNB3	EMV Discretionary Data Token
EMV-ISS-SCRIPT-RSLTS-TKN	BJ	TKNBJ	EMV Issuer Scripts Results
EMV-RQST-TKN	B2	TKNB2	EMV Request Data Token
EMV-RESP-TKN	B5	TKNB5	EMV Response Data Token
EMV-SCRIPT-TKN	B6	TKNB6	EMV Script Data Token
EMV-STAT-TKN	B4	TKNB4	EMV Status Token
ISSUER-FEE-REBATE-TKN	30	TKN30	Issuer Fee Rebate Token
MICR-DATA-TKN	12	TKN12	MagneticInk Char Recognition
MULT-CRNCY-TKN	BD	TKNBD	Multi-Currency Token
MULT-LN-TKN	BK	TKNBK	Multiple LN Token
NAM-TKN	08	TKN08	Customer Short Name Token
ORIG-CRNCY-60-TKN	BE	TKNBE	Original Currency 60 Token
PRISM-TKN	28	TKN28	Prism Token
PSEUDO-CRD-NUM-TKN	BL	TKNBL	Pseudo Card Number Token
RVSL-DAT-TIM-TKN	BH	TKNBH	Reversal Date Time Token
SURCHARGE^DATA^TKN	25	TKN25	Surcharge Data Token
SWI-TKN	B0	TKNB0	Acquirer Generic Switch TKN

Table 8-4 (Cont.) BASE - DDLBATKN

TOKEN NAME	ID	NAME	DESCRIPTION
SWI-TKN	B1	TKNB1	Issuer Generic Switch TKN
TLF-TKN	B7	TKNB7	Transaction Log FileName TKN
TRACK1-TKN	23	TKN23	Track1 Token
TRK3-TKN	BG	TKNBG	Track3 Token
TXN-DESCR-TKN	B9	TKNB9	Transaction Descrip Token
TXN-PRFL-TKN	B8	TKNB8	Transaction Profile Token
TXN-SUBTYP-TKN	BM	TKNBM	Transaction Subtype Token

Table 8-5 ATM - DDLATTKN

TOKEN NAME	ID	NAME	DESCRIPTION
ADDL-HOPR-TKN	22	TKN22	Additional Hopper Token
AT-FLG1-TKN	24	TKN24	ATM Flag1 (Misc fields) TKN
AT50-TKN	03	TKN03	BASE24-atm Release 5.0 Token
ATM-BAL-TKN	AB	TKNAB	BASE24-atm Balances Token
CASH-ACCPT-TERM-SETL-TKN	AD	TKNAD	Cash Accept Term Setl Token
ICHG-COMPLIANCE-ATM-TKN	A6	TKNA6	Interchange Compliance Token
MBC-BD-TKN	A8	TKNA8	Merch Bank Cntr Bag Deposit
MBC-MX-TKN	A9	TKNA9	Merch Bank Cntr Money Exchg
MBC-SETL-TKN	AA	TKNAA	MBC Settlement Token
NCD-TKN	A5	TKNA5	Non-Currency Dispense Token
PINC-TKN	06	TKN06	PIN Change Token
PS2000-ATM-TKN	21	TKN21	Payment service 2000 ATM TKN
SM-PRI-TKN	A0	TKNA0	Smart Card Primary Token
SM-REFR-TKN	A2	TKNA2	Smart Card Refresh Token
SM-TERM-SETL-TKN	A4	TKNA4	Smart Card Terminal Sttlmnt
SM-VISA-TKN	A3	TKNA3	Smart Card Visa Token
SSBB-TKN	07	TKN07	Self-Service Bank Base Token
SSBC-TKN	14	TKN14	Self-Service Bank Chk Token

Table 8-5 (Cont.) ATM - DDLATTKN

TOKEN NAME	ID	NAME	DESCRIPTION
SSBC-TERM-SETL-TKN	15	TKN15	Self-Serv Bank Chk Term Setl
STMT-PRNT-TKN	02	TKN02	Statement Print Token
MULT-ACCT-TKN	A7	TKNA7	Multiple Account Token

Table 8-6 POS - DDLPSTKN

TOKEN NAME	ID	NAME	DESCRIPTION
ACH-DB-TKN	11	TKN11	Auto Clearing House Debit
ADDR-VER-TKN	01	TKN01	Address Verification TKN
ALT-MERCH-ID	16	TKN16	Alternate Merchant ID Token
AMEX-TKN	10	TKN10	American Express Token
AUTHN-DATA-TKN	CE	TKNCE	Authentication Data Token
CERT-TKN	C3	TKNC3	Certificate Token
CHK-AUTH-TKN	05	TKN05	Check Authorization Token
CHK-AUTH2-TKN	29	TKN29	Check Authorization Token
CHK-CALLBACK-TKN	31	TKN31	Check Callback Token
CRDHLDR-SERIAL-NUM-TKN	C8	TKNC8	Cardholder Serial Number TKN
DUKPT-DATA-TKN	CA	TKNCA	Derived Unique Key Per Trans
EBT-AVAIL-BAL-TKN	U1	TKNU1	EBT Available Balance Token
EBT-VOUCHER-NUM-TKN	U0	TKNU0	EBT Voucher Number Token
IAVS-DATA-TKN	CF	TKNCF	IAVS Data Token
ICHG-COMPLIANCE-TKN	20	TKN20	Interchange Compliance Token
MHI-ADDL-DATA-TKN	C6	TKNC6	Merchant Host Interface Additional DataToken
MRCH-SERIAL-NUM-TKN	C9	TKNC9	Merchant serial Number Token
OPT-DATA-TKN	C5	TKNC5	Increased Optional Data TKN
POS-BAL-TKN	CB	TKNCB	POS Balances Token
POS-DATA1-TKN	CH	TKNCH	POS Data1 Token
POS-MRCH-TKN	CI	TKNCI	POS Merchant Token
PS2000-OFFL-TKN	19	TKN19	VISA Pmt Serv 2000 Offline
PS2000-TKN	17	TKN17	VISA Payment Service 2000
PS50-TKN	04	TKN04	POS 5.0 Token

Table 8-6 (Cont.) POS - DDLPSTKN

TOKEN NAME	ID	NAME	DESCRIPTION
PS51-TKN	C0	TKNC0	POS 5.1 Token
PT-SRV-DATA-TKN	C4	TKNC4	Point of Service Data Token
PURCHASE-TKN	C2	TKNC2	Purchasing Card & Fleet Card
STA-ID-TKN	C1	TKNC1	Station ID Token
STORED-VALUE-TKN	U2	TKNU2	Stored Value Token
TRANS-STAIN-XID-TKN	C7	TKNC7	Unique Transaction Identifier

9

Managing and Monitoring

Learn how to manage tasks and trails, and how to monitor processes, which includes analyzing the process report, viewing record counts, and handling errors.
This topic includes the following:

Managing Tasks

Tasks are processes that are special runs, such as a one-time data synchronization, or direct file extraction. Tasks are useful in managing Oracle GoldenGate, because they allow you to load data that may have been missed due to a variety of system errors. You can define a task with the GGSCI commands:

```
GGSCI> ADD EXTRACT group_name, SOURCEISTABLE
GGSCI> ADD REPLICAT group_name, SPECIALRUN
```

When you define a task, you must include the task type parameter in the parameter file. For the Extract parameter file, include `SOURCEISTABLE` or `SOURCEISFILE`. For the Replicat parameter file include `SPECIALRUN`.

Manager can purge tasks. To purge tasks enter parameters such as:

```
PURGEOLDTASKS EXTRACT wildcard_spec, AFTER number HOURS, USESTOPSTATUS
PURGEOLDTASKS REPLICAT wildcard_spec, AFTER number DAYS, USESTOPSTATUS
PURGEOLDTASKS ER wildcard_spec, AFTER number HOURS, USESTOPSTATUS
```

Getting Information on Tasks

You can retrieve information about a task using the `INFO` and `STATUS` commands with the `TASKS` or `ALLPROCESSES` options:

```
GGSCI> INFO EXTRACT *, TASKS
GGSCI> INFO REPLICAT *, ALLPROCESSES
GGSCI> STATUS ER *, ALLPROCESSES
```

`TASKS` reports on `SPECIALRUN` or `SOURCEISFILE` tasks. `ALLPROCESSES` reports on all processes.

Managing Tasks Using the Process Name

Tasks defined with `SPECIALRUN`, `SOURCEISFILE`, or `SOURCEISTABLE` do not require a group name. Even without this name, it is possible to communicate with these running tasks by using the `SEND PROCESS` command. The syntax for this uses the process name instead of a group name as shown below.

```
GGSCI> SEND PROCESS process_name {text | WAKE | BREAK}
```

The *`text`* option can be any one of the subset of GGSCI commands that are recognized by the receiving process.

See SEND PROCESS for more details.

Managing Oracle GoldenGate Trails

Oracle GoldenGate trails can be managed by allocating optimal storage for the trail files and setting parameters for cleaning up trail files that are no longer needed.

Initial Allocation of Storage for Trails

To prevent trail activity from interfering with business applications, use a separate disk managed by a disk process different than that of the application.

To ensure there is enough disk space for the trail files, follow these guidelines:

- For trails on the source system, there should be enough space to handle data accumulation if the network connection fails. In a failure, reading from a trail terminates but the primary Extract group reading from logs or audit file continues extracting data. It is not good practice to stop the primary Extract group to prevent further accumulation. The logs could recycle or the audit files could be off-loaded.
- For trails on the target system, data will accumulate because data is extracted and transferred across the network faster than it can be applied to the target database.

To estimate the required trail space

1. Estimate the longest time that you think the network can be unavailable.
2. Estimate how much transaction log volume you generate in one hour.
3. Use the following formula:

```
trail disk space =
transaction log volume in 1 hour x number of hours down x .4
```

> **Note**
>
> The equation uses a multiplier of 40 percent because Oracle GoldenGate estimates that only 40 percent of the data in the transaction logs is written to the trail.

A more exact estimate can be derived by either:

4. Configuring Extract and allowing it to run for a set time period, such as an hour, to determine the growth. This growth factor can then be applied to the maximum down time.

Plan to store enough data to withstand the longest anticipated outage possible because you will need to re-synchronize the source and target data should the outage outlast the disk capacity.

Ongoing Trail Management

Oracle GoldenGate provides options that let you manage your trails in two ways.

- Based on the number and size of the files.

  The `MEGABYTES, MAXFILES`, and `EXTENTS` options specify how large each trail file may become, and how many files may exist before Extract stops with an error.

- With the PURGEOLDEXTRACTS parameter.

  This lets you purge old extracted data you no longer need. This can be based on rules you set up.

  – The MINKEEPHOURS, MINKEEPDAYS options set the time to keep files. MINKEEPFILES sets the minimum number of files to keep.

  – In the Manager only, the USECHECKPOINTS option uses checkpoints to determine whether processing is complete. You can also set the CHECKMINUTES parameter to control how often the process checks the parameters to determine if anything must be purged.

Setting the Size of the Trail

Two options for managing trail size are MEGABYES and MAXFILES. MEGABYTES lets you specify how large your trail file gets before your data rolls to another trail file. It is useful if you want to equally distribute data between your files. The default size is 134 megabytes and the largest size supported is two gigabytes. MAXFILES lets you specify the number of trail files Oracle GoldenGate creates. The default is 100 files. Allowing multiple files lets data roll over when one file is full, which prevents errors. The syntax for using MEGABYTES and MAXFILES is:

```
GGSCI> ADD EXTTRAIL trail_name, EXTRACT group_name, MEGABYTES num, MAXFILES num
```

Trails that either reside on the local node or on a node that is connected by Expand are considered local for NonStop. For these trails, you can also control size by setting the files' primary, secondary and maximum number of extents. The syntax for this is:

```
GGSCI> ADD EXTTRAIL trail_name, EXTRACT group_name
[, EXTENTS (primary, secondary, max)]
```

The defaults for EXTENTS are (64, 128, 512).

From GGSCI, an INFO of the trail will show the current trail settings.

Example 9-1 Showing Trail Settings

```
GGSCI> INFO EXTTRAIL GGSDAT.ET
Extract file: \NY.$DATA04.GGSDAT.ET
       Extract group: EXTSQL
               Owner: 150,110
            Security: NUNU
       Current seqno: 0
         Current rba: 2280
     Primary extent: 64
   Secondary extent: 128
        Max extents: 512
          Max files: 100
```

Setting the PURGEOLDEXTRACTS rules

You can set PURGEOLDEXTRACT in the Manager, Extract, or Replicat, but only Manager has options.

In the Manager

You can set options for purging trails with the PURGEOLDEXTRACTS in the Manager's parameter file.

- Use USECHECKPOINTS to purge when all processes are finished with a file as indicated by checkpoints. This is the default, but it can be turned off with the NOUSECHECKPOINTS option.

- MINKEEPHOURS or MINKEEPDAYS to keep files n hours or days. MINKEEPFILES to keep at least n files including the active file. The default is 1.

Only *one* of the three MINKEEP options should be set. If more than one is entered the system will select one based on the following:

- If both MINKEEPHOURS and MINKEEPDAYS are specified, only the last setting will be used and the other will be ignored.
- If both MINKEEP{HOURS|DAYS} and MINKEEPFILES are specified MINKEEP{HOURS|DAYS} will be used and MINKEEPFILES will be ignored.

In Extract or Replicat

You cannot set any options for the Extract and Replicat PURGEOLDEXTRACTS parameter. In this case the trail is purged as soon as the process moves to the next trail.

Manager Purge Trail Processing

If PURGEOLDEXTRACTS is set in the Manager parameter file, when the Manager reaches CHECKMINUTES the purge rules are evaluated as explained below.

1. USECHECKPOINTS only

   If there are no minimum rules set with the USECHECKPOINTS option, MINKEEPFILES defaults to 1. If checkpoints indicate that a trail file has been processed, it will be purged unless it would fall below this one file minimum.

2. USECHECKPOINTS with MINKEEP rules

   If checkpoints indicate that a trail file has been processed, it will be purged unless doing so would violate the applicable MINKEEP{HOURS|DAYS} or MINKEEPFILES rules. These PURGEOLDEXTRACTS minimum rules are set as explained in Setting the PURGEOLDEXTRACTS rules.

3. NOUSECHECKPOINTS only

   If there are no minimum rules and checkpoints are not to be considered, the file will be purged, unless doing so will violate the default MINKEEPFILES of 1.

4. NOUSECHECKPOINTS with MINKEEP rules

   The file will be purged unless doing so will violate applicable MINKEEP{HOURS|DAYS} or MINKEEPFILES rules. Refer to Setting the PURGEOLDEXTRACTS rules for information on setting these PURGEOLDEXTRACTS minimum rules.

Example 9-2 Purge Processing Examples

- Trail files AA000000, AA000001, and AA000002 exist. The Replicat has been down for four hours and has not completed processing any of the files

  The Manager parameters include:

  ```
  PURGEOLDEXTRACTS $DATA1.DB.AA*, USECHECKPOINTS, MINKEEPHOURS 2
  ```

  Result: The time files that are not accessed must be retained has been exceeded. No files will be purged, however, because checkpoints indicate that the files have not been fully processed by Replicat.

- Trail files AA000000, AA000001, and AA000002 exist. The Replicat has been down for four hours and has not completed processing.

  The Manager parameters include:

```
PURGEOLDEXTRACTS $DATA1.DB.AA*, NOUSECHECKPOINTS, MINKEEPHOURS 2
```

Result: All trail files will be purged since the minimums have been met.

- The following is an example of why only one of the MINKEEP options should be set

  Replicat and Extract have completed processing. There has been no access to the trail files for the last five hours. Trail files AA000000, AA000001, and AA000002 exist.

  The Manager parameters include:

  ```
  PURGEOLDEXTRACTS $DATA1.DB.AA*, USECHECKPOINTS, MINKEEPHOURS 4, MINKEEPFILES 4
  ```

  Result: USECHECKPOINTS requirements have been met so the minimum rules will be considered when deciding whether to purge AA000002.

  There will only be two files if AA000002 is purged, which will violate the MINKEEPFILES parameter. Since both MINKEEPFILES and MINKEEPHOURS have been entered, however, MINKEEPFILES is ignored. The file will be purged because it has not been modified for 5 hours, which meets the MINKEEPHOURS requirement of 4 hours.

  The Manager process determines which files to purge based on the Extract processes configured on the local system. If at least one Extract process reads the trail file, Manager applies the specified rules.

  For more information see, PURGEOLDEXTRACTS for Extract and Replicat and ADD EXTTRAIL.

Recommendations for Managing Trail Purges

Consider the following recommendations for managing Oracle GoldenGate trails.

- For setting the purge rules, it is recommended that: you
  - Specify PURGEOLDEXTRACTS in the Manager parameter file so you manage your trails from a single location.
  - Purge trail files through Extract or Replicat only when one process is processing the trail, such as a data pump. Use Manager to purge trail files that are being processed by both Extract and Replicat.
  - Use USECHECKPOINTS to ensure that the checkpoints of both Extract and Replicat are considered and reduce the chance of data loss.
  - Be aware that PURGEOLDEXTRACTS in Extract or Replicat can remove trails still needed by the Coordinator. If you use the Coordinator, specify PURGEOLDEXTRACTS in the Manager to manage the Coordinator checkpoints.
- The rules should be assigned to the process that resides where the trail must be cleaned.

  For example, if there are three nodes: \A where the Extract is running and extracting the data; \B where a subset of the data is replicated and \C where another part of the data is replicated, it is the Manager on \A that should be assigned the parameters that define how to manage the trails.

  For USECHECKPOINTS, this Manager will need to know the location of the checkpoint files on \B and \C, but this can be accomplished with REMOTECHKPT as shown below.

  ```
  GGSCI> ADD REMOTECHKPT \node.$volume.subvolume.REPCTXT
  ```

Oracle GoldenGate Self Describing Trail Files

The default behavior is to store and forward metadata from the source to the target and encapsulates it in each of the trail files.

Metadata records are as follows:

- *Database Definition Record* (DDR)

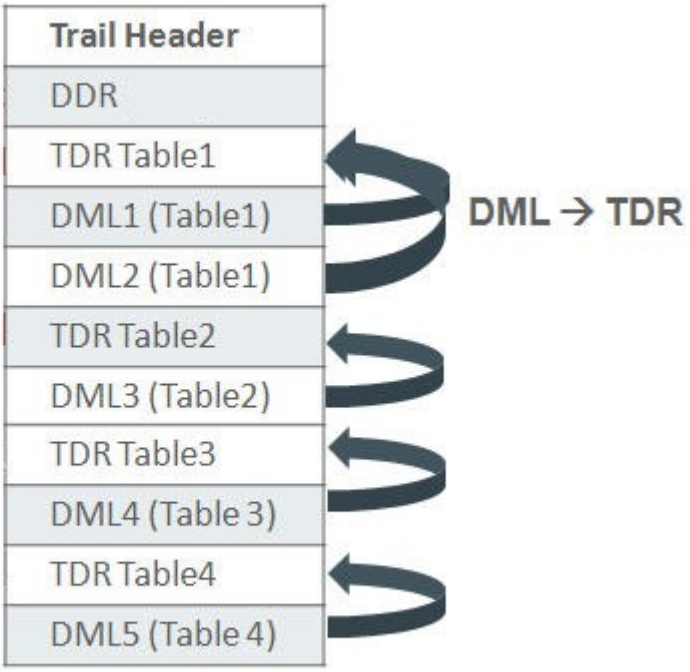
  A DDR provides information about the specific database, such as character set and time zone. Extract writes a DDR to the trail following the file header to store the database metadata for the source database.

- *Table Definition Record* (TDR)

  A TDR provides details about the definition about a table and the columns that it contains. The content of this record is similar, though not identical, to a record in a `sourcedef` file that was created using `DEFGEN`. Extract writes a new TDR when the output trail rolls over to a new file or the source table definition has changed.

  The metadata records in a self-describing trail file format operate as follows:

The metadata records in a self-describing trail file format operate as follows:



Using self-described trail files eliminates the need for `SOURCEDEFS` and `ASSUMETARGETDEFS` so parameter files are simpler and it is easier to configure. This feature also provides:

- A reduction in trail file size due to object name compression.
- No necessity to create and maintain source definitions files.

- Replicating OpenSys databases to HP-NonStop no longer requires TARGETDEFS and the mapping of column names, as well as mapping ANSI names to three part Tandem names.
- No necessity to create and maintain source definitions files.

Understanding the Self-Describing Trail Behavior

OpenSys (Non Enscribe or SQL/MP) source databases

When performing table maintenance on objects that are part of an extract group only after the remote Extract has completely output all the data change records to the trail and the Extract process is stopped. Then after both the source and target changes are completed, restart the Extract. There is no need to regenerate TARGETDEFS using DEFGEN on NSK and moving them to OpenSys, this assumes that since it is not required to map the OpenSys table names to NSK (Tandem) names or colmap individual columns, that all of it was already removed from the Extract parameter file. In the case of adding a new table, if the parameter files already wildcards this name, simply create the target table and then the source.

SQL/MP to SQL/MP

If replicating new columns, then no action is required to update metadata. For Drop and Create table statements, you must perform those database actions on the target and then the source outside of Oracle GoldenGate replication, but no other action is required. This assumes that the qualified table names are either, already in the parameter file, or fit a wildcard specification.

Enscribe to Enscribe

Logger Capture

The logger process has not been changed and will continue write trails without a file header or metadata this is equivalent to format release 9.5. There is no means to change this; therefore, Oracle recommends that if you are not already using Extract to pump logger trails, you should start using them. The Extract pump needs the source DICTIONARY and each file statement needs a DEF or TARGETDEF option on the FILE parameter to create metadata.

TMF Extract Capture

The TMF based Extract writes the metadata based on a provided DICTIONARY and specific DEF or TARGETDEF option on the FILE parameter.
If you have not provided the record DEF for a file, then a fabricated metadata record is generated. There is an info message in Extract as follows:

```
2018-08-07 04:44:50 OGG INFO 103 No columns loaded for
\NODE.$DISK.SUBVOL.FILE, fabricating metadata TDR record.
```

Regardless of the method of capture, you can only use fabricated Enscribe metadata, when Extract or Replicat has No Colmaps, Filters, @functions(), or anything else that requires a real column information. Typically, you should use the fabricated metadata only when the source and target are the same files and no DEF or TARGETDEFS is used. Otherwise attempting to do so causes an abend.

In the following example, the source metadata was fabricated, while the target has a TARGETDEFS, it abends attempting to colmap default names.

```
MAP \NODE.$DISK.SUBVOL1.FILE, target \NODE.$DISK.SUBVOL2.FILE, targetdef REC-DEF;
Metadata loaded from trail for file \NODE.$DISK.SUBVOL1.FILE
Building Map ID 1 for \NODE.$DISK.SUBVOL1.FILE
2018-08-07 04:44:49 OGG WARNING 101 Source metadata found in trail for
\NODE.$DISK.SUBVOL1.FILE was fabricated and may not match the target
\NODE.$DISK.SUBVOL2.FILE.
```

```
Error with default mapping. No matching fields found in source and target
2018-08-07 04:44:50 OGG ERROR    101 Error in COLMAP clause from compile_map.
2018-08-07 04:44:50 OGG ERROR    191 REPLICAT abending.
```

Managing Log Trails

Unlike trails that are created externally, if a trail created by Logger runs out of space there are no audit records to be reprocessed once the problem is fixed. This can cause data loss, so it is important to have adequate space available for log trails. The following steps help do this.

- Include adequate trail space when the Logger process is added.

  Log trail files are pre-allocated during `ADD LOGGER` so this ensures that the space is available before the logging process begins.

- Monitor and adjust the trail space as needed.

  During processing, adjust the number and size of each log trail as needed by editing the Logger parameter file with the command `EDIT PARAM LOGPARM`. Then the number of files can be increased or decreased by changing the `NUMFILES` option, and the size of each trail file can be adjusted by changing the `MEGABYTES` or `EXTENTS`. The changes will be activated when an `ALTER LOGGER` command is issued.

- Monitor the impact of the trail space on your system.

  Manage creates the next log trail file if it is not available when it is time to rollover. This helps ensure that data will not be lost, but it also means that you may have more log trail files than specified in the `NUMFILES` of `LOGPARM`.

Monitoring Processing

You can monitor the state of Oracle GoldenGate processing with the following tools.

Tool	Description
The event log	The event log, named `LOGGGS`, shows processing events, messages, errors, and warnings generated by Oracle GoldenGate. You can view the event log from GGSCI using the `VIEW GGSEVT` command.
Process reports	Oracle GoldenGate generates a report for Extract, Replicat, and Manager at the end of each run. The report provides information about run-time parameters and statistics, including process events and operations that were performed.
N/A	The name of a process report is either `Manager` for the Manager process, or for Extract and Replicat, is the same as the group name. For example, a report for an Extract group `EXTORD` would have a report named `GGSRPT.EXTORD`. You can view the process report, save and store it, and use it for generating run-time statistics.
Record counts	You can produce record counts at designated times during processing by using the `REPORTCOUNT` parameter in the Extract and Replicat parameter files. Results are printed to the process report file and to the screen.
Discard files	The discard file logs information about operations that failed. To generate discard files, use the `DISCARDFILE` parameter in the Extract or Replicat parameter file. The SQL formatted discard file logs information and SQLCI formatted statements for operations that failed replication to SQL/MP. To generate, use the `SQLFORMATDISCARDFILE` parameter in the Replicat parameter file. To control how discard files are generated, use the `DISCARDROLLOVER` parameter in the Extract or Replicat parameter file or the `SQLFORMATDISCARDROLLOVER` in the Replicat parameter file. These parameters have provisions for specifying when new files are created.

Tool	Description
GGSCI INFO commands	Using the GGSCI commands INFO, SEND, STATUS, and STATS, you can retrieve information on Oracle GoldenGate processes.

Each of these tools is discussed in greater detail below.

Error Handling

There are several error handling parameters available in Oracle GoldenGate. In addition, Oracle GoldenGate provides error-handling options for Replicat and TCP/IP processing.

Error Handling Parameters

Error handling parameters let you insert missing records, prevent duplicate records from being loaded. For a complete list, see Oracle GoldenGate Parameters.

Handling Replicat Errors

To control the way that Replicat responds to errors, use the REPERROR parameter in the Replicat parameter file. This parameter handles most errors in a default fashion (for example, to cease processing), and you can specify REPERROR options to handle other errors in a specific manner, or to ignore them altogether.

REPERROR provides the following options:

Option	Description
ABEND	Roll back the Replicat transaction and terminate processing. This is the default.
DISCARD	Log the error to the discard file but continue processing the transaction and subsequent transactions.
EXCEPTION	Treat the error as an exception. To handle an exception, create an entry with the MAP parameter that runs after the error. For example, you can map a failed update statement to an exceptions table dedicated to missing updates.
FILEOP	Set the error handling for a particular I/O type.
IGNORE	Ignore the error.
RESET	Remove all REPERROR settings made at the root level of the parameter file above the RESET parameter.
RETRYOP	Retry the operation. Use the MAXRETRIES argument with RETRYOP to specify the number of times to retry an operation. To control the interval between retries, use the RETRYDELAY parameter.
RETRYOPEN	Retry a file open error.
TRANSABORT	Abort the current target transaction and then retry it.

TCP/IP Error Handling

The TCPERRS file in the Oracle GoldenGate installation location contains preset TCP/IP errors and instructions for how Oracle GoldenGate generally reacts to them. If a response is not explicitly defined in this file, Oracle GoldenGate responds to TCP/IP errors by exiting.

> **Note**
>
> The Manager process is an exception. When Manager has an IP error it retries every 60 seconds and does not abend. It does not use the TCPERRS file to determine the number of retries or the delay.

Example 9-3 Sample of the TCPERRS File

```
#
# TCP/IP error handling parameters
# Default error response is abend
#
# error       Response  Delay (csecs) Max Retries
ECONNABORTED  RETRY     1000          10
ECONNREFUSED  RETRY     1000          12
ECONNRESET    RETRY     500           10
ENETDOWN      RETRY     3000          50
ENETRESET     RETRY     1000          10
ENOBUFS       RETRY     100           60
ENOTCONN      RETRY     100           10
EPIPE         RETRY     500           10
ESHUTDOWN     RETRY     1000          10
ETIMEDOUT     RETRY     1000          10
NODYNPORTS    RETRY     100           10
```

Altering TCP/IP Error Handling Parameters

To alter the instructions or add instructions for new errors, open the file in a text editor and change any of the values in the following columns:

- Error column: Specifies a TCP/IP error for which you are defining a response.
- Response column: Controls whether Oracle GoldenGate tries to connect again after the defined error.
- Delay column: Controls how long Oracle GoldenGate waits before attempting to connect again.
- Max Retries column: Controls the number of times that Oracle GoldenGate attempts to connect again before aborting.

See Oracle GoldenGate Parameters for details about the TCP/IP error messages, their causes, effects, and recovery.

Using Discard Files

Two types of discard files are available with Oracle GoldenGate for HP NonStop:

- Discard files identify a record and operation that failed., providing information to help troubleshoot the error.

- SQL discard files provide SQLCI formatted input to process the operation to an SQL/MP database once the error is corrected.

Using the Discard File

The discard file logs information about operations that failed. Extract discard records have a header and a data portion; the discard file is entry-sequenced. Replicat produces discard records in an external, easy-to-understand format

Full record images are provided when `IO-TYPE` is `Delete`, `Insert` or `Update`. Each record has the same format as if retrieved from a program reading the original file or table directly. For SQL tables, datetime fields, nulls and other fields are output exactly as a program would `SELECT` them into an application buffer. Even though datetime fields are represented internally as an eight byte timestamp, their external form can be up to 26 bytes expressed as a string. Enscribe records are retrieved as they exist in the original file.

Full record images are output unless the original file has the `AUDITCOMPRESS` attribute set to `ON`. When `AUDITCOMPRESS` is `ON`, compressed update records are generated whenever the original file receives an update operation. (A full image can be retrieved by Extract using the `FETCHCOMPS` parameter.)

When the operation type is `Insert` or `Update`, the image is the contents of the record *after* the change is made. When the operation type is `Delete`, the image is the contents of the record *before* the change.

To control how discard files are generated, use the `DISCARDROLLOVER` parameter in the Extract or Replicat parameter file. The parameter has provisions for specifying when new files are created.

Generating Discard Files

To generate a discard file, use the `DISCARDFILE` parameter in the Extract or Replicat parameter file.

If there is no `DISCARDFILE` parameter for Replicat, a discard file is created by default when Replicat is started from GGSCI (as opposed to when it is run from the TACL prompt.) The default discard file will have the following characteristics:

- The file name is derived by appending a D to up to 7 characters of the Replicat group name. For example, the Replicat `GROUPA` will create discard file `GROUPAD` and Replicat `GROUPAB` will create discard file `GROUPABD`. If the Replicat group name is more than 7 characters, the process name is used instead. Replicat `GROUPABC` with process name `$ABR00` will create discard file `ABR00D`.
- The file is created in the report file location.
- The extents are set to (4, 4, 100)

> **Note**
>
> Discard files that have been created by default cannot be rolled over.

To specify a non-default location or characteristic use the `DISCARDFILE` parameter in the Replicat parameter file.

Using the SQL Formatted Discard File

SQL discard files provide SQLCI formatted input for failed replication operations to a target SQL/MP database. Specify `SQLFORMATDISCARDFILE` in the Replicat parameter file to generate the SQL discard file.

To use the SQL formatted discard file the target must be SQL/MP. The source of the operation can be any database and platform that Oracle GoldenGate supports.

The SQLCI formatted input will only process DML records once the original problem has been corrected.

To control how SQL formatted discard files are generated, use the parameter `SQLFORMATDISCARDROLLOVER`. This parameter specifies when new files are created.

SQL Discard File Example

The following is an example of a section of a SQL formatted discard file record:

```
SET SESSION ERROR ABORT ON;
BEGIN WORK;
-- Problem with Insert Record, Seqno 0, RBA 1234 Error -8227
INSERT INTO \NY.$DATA2.SALES11.CUSTOMER & (CUST_CODE, NAME, CITY, STATE) VALUES &
(“A543", “Advantage Software            ",&
“SEATTLE            “, “WA");
COMMIT WORK;
```

The following is an example of the corresponding section from the discard file record:

```
*** ERROR from SQL [-8227]: The row being inserted or updated in underlying
*** table \NY.$DATA2.SALES11.CUSTOMER contains a key value that must be
*** unique but is already present in a row.
Error -8227, guardian 10 occurred with insert record (target format). . .
*---  2013-08-28 09:46:13.775946  ----*
0,    4:CUST_CODE = ‘A543'
1,   12:NAME = ‘Advantage Software'
2,   46:CITY = ‘SEATTLE'
3,   70:STATE = ‘WA'
*--- End Discard Record --- *
Process Abending: 2013-08-28 09:46:14
```

Editing the SQL Discard File

The SQL discard file is an edit file that can be changed if needed. To change the file, first do a rollover on the file to trigger Replicat to close it and begin to use a new one. Then edit the file.

When you edit the SQL formatted discard file input, you must make sure your statements are valid for SQLCI:

- Commands greater than 132 bytes must force a line break.
- Strings must be enclosed in quotation marks. If embedded quotes are part of the data, the outermost set of quotation marks must be of the opposite type (single or double.)
- Quoted strings longer than the maximum size must be broken into the 132 byte segments surrounded by quotation marks and terminated by the (&) continuation symbol. Note that the quotation marks and & symbol are all counted in the 132 byte limit.
- The requirements for date-time and timestamp syntax are specific to the metadata.

Conflict Detection with SQLEXEC

SQLEXEC works on SQL for NonStop databases to call queries you specify. This lets you leverage application rules to resolve conflicts between incoming source records. To do this, add a SQLEXEC parameter in a MAP statement in the Replicat parameter file.

Some applications that support distributed processing provide their own methods for handling conflicts. Such methods include IP persistent routers or application privileges that prevent multiple users from modifying the same data. These rules can be combined with the use of SQLEXEC procedures.

A SQLEXEC Example

The following is an example of basic conflict detection based on a timestamp. It raises an exception based on a user-defined error number passed to Replicat and a SQL query to check the value of the TIMESTAMP column. If the timestamp is newer than the one on the existing record, the update is performed. Otherwise, the operation is ignored (because the newer timestamp is considered more valid) and a message is generated to the Replicat process report.

Example 9-4 REPLICAT Parameter File for Conflict Detection

```
REPERROR (9999, EXCEPTION)
MAP $DATA.MASTER.SRC, TARGET $DATA.MASTER.TAR,
SQLEXEC (ID check, QUERY " SELECT TIMESTAMP FROM TARGTAB"
WHERE PKCOL =?P1 ", ERROR IGNORE);
PARAMS (P1 = PKCOL)),
FILTER (CREATED_BY <> "DBA"),
FILTER (ON UPDATE, BEFORE.TIMESTAMP < CHECK.TIMESTAMP,
RAISEERROR 9999);

INSERTALLRECORDS
MAP $DATA.MASTER.SRC, TARGET $DATA.MASTER.TAREXEC,
EXCEPTIONSONLY,
COLMAP (USEDEFAULTS, ERRTYPE = "UPDATE FILTER FAILED.");
```

In the example, the query is run under the logical name of check. Values retrieved from this query can be utilized anywhere in the MAP statement by referencing check.column name.

The FILTER statements in the example parameter file are processed in the order that they are written. If, in the first FILTER statement, the value of the CREATED_BY column in the record being applied by Replicat is equal to the DBA account, the operation is accepted for processing by the second FILTER statement. Otherwise, it is ignored.

In this example, SQLEXEC also detects database errors, but ignores them and continues processing. This is the default action for ERROR.

```
SQLEXEC (ID check, QUERY " SELECT TIMESTAMP FROM TARGTAB"
WHERE PKCOL =?P1 ',ERROR IGNORE);
```

However, SQLEXEC could perform any of the following:

Syntax	Description
ERROR REPORT	Write the database error to a report.

Syntax	Description
`ERROR RAISE`	Enable the same error handling capabilities available for table replication errors.
`ERROR FINAL`	Enable the same error handling capabilities as `ERROR RAISE`, but also ignore any further queries left to process.
`ERROR FATAL`	Abend the process immediately.

In the second `FILTER` statement, the `ON UPDATE` clause directs the filter to run only for update statements. It compares the value of `BEFORE.TIMESTAMP` (the timestamp of the row that Replicat is attempting to apply) to `CHECK.TIMESTAMP` (the timestamp of the row already in the database). If the row in the database is newer than the row being applied, then the filter raises an error and the update is ignored.

In the example, the error correction was implemented with `RAISEERROR` in the `SQLEXEC` clause in the first `MAP` statement, but it could have been implemented in the second `MAP` statement by replacing the `COLMAP` clause with a `SQLEXEC` clause.

To handle specific issues, additional `SQLEXEC` statements could be performed after the filter or even between the filter statements for increased control.

Using the Event Log

The Oracle GoldenGate event log shows processing events, messages, errors, and warnings generated by Oracle GoldenGate. Although this information is also recorded in the NonStop Event Management System (EMS), viewing the Oracle GoldenGate log is sometimes more convenient. Use GGSCI `VIEW GGSEVT` command to view the event log.

Using the Process Report

Oracle GoldenGate generates a report about Manager, Logger, Extract, Replicat, and Syncfile at the end of each run. The report provides information about run-time parameters and statistics, including process events, and operations that were performed. The name of a process report is either `MANAGER` for the Manager process, or it is the same as the group name for Extract and Replicat. By default, reports are created in the subvolume `GGSRPT`. For example, a report for an Extract group `EXTORD` would have a report named `GGSRPT.EXTORD`.

Generate process reports with the `SEND EXTRACT` *`group_name`* command and the following options:

Report	Option	Description
The end of an audit trail	`AUDITEND`	Queries Extract to determine whether all records in the TMF audit trails have been processed. This command indicates whether more Extract or Replicat activity must occur before a scheduled switch between databases. Until `AUDITEND` returns "All audit processed," more data must be processed before it can be assumed that secondary databases are synchronized.
Processing status	`STATUS`	Returns a detailed status of the processing state, including current position and activity.

Report	Option	Description
Processing statistics	REPORT	Generates an interim statistical report to the report file, including the number of inserts, updates, and deletes since the last report (default) or according to report options that can be entered. For more information, see Send Report.
TCP/IP statistics	GETTCPSTATS RESETTCPSTATS	Retrieves TCP/IP statistics, such as the quantity and byte length of inbound and outbound messages, the number of messages received and sent, wait times, process CPU time, and byte transmit averages. Time accumulates when Extract is waiting on a socket send or receive and all times are reported in microseconds. Resets the TCP/IP statistics so the next report displays fresh statistics

> **Note**
>
> In Oracle GoldenGate for NonStop, several additional reporting options are available. For specifics, see Report commands.

Example 9-5 Sample Report

```
RMTTRAIL $DATA10.LOGGER.R1000038, RBA 5348453
Session Index 1
Stats started 2011/01/10 11:46:18.804165 0:00:41.522086
Local address 192.0.2.2:1000 Remote address 192.0.2.2:1000
Inbound Msgs 199 Bytes 2337, 57 bytes/second
Outbound Msgs 200 Bytes 5389492, 131451 bytes/second
Recvs 199
Sends 200
Avg bytes per recv 11, per msg 11
Avg bytes per send 26947, per msg 26947
Recv Wait Time 17592208, per msg 88403, per recv 88403
Send Wait Time 774603, per msg 3873, per send 3873
Process CPU Time 0:00:07.715372
```

Viewing Process Reports

To view a process report, view the file directly from the operating system's command shell, or use the `VIEW REPORT` command in GGSCI. You also can view process reports from the Activity Console by clicking `More Info` beneath the name of the process.

Storing Process Reports

By default, process reports are stored in the `GGSRPT` subvolume of the Oracle GoldenGate installation volume. You can designate an alternative file name by using the `REPORT` option of the `ADD EXTRACT` and `ADD REPLICAT` commands when you create the group from the GGSCI interface. Specify the fully qualified file name.

Once paired with Extract or Replicat, the report file may remain in its original location, or you may change its location with the `ALTER` command, such as:

```
ALTER EXTRACT group_name REPORT filename
```

Managing Process Reports

Whenever a process starts, a new report file is created, and a sequence number is appended to the name of the old file. The naming sequence goes from no sequence number (current), to 0 (the most recent) to 9 (the oldest), for example: `$DATA.GGSRPT.EXTORD, $DATA.GGSRPT.EXTORD0, $DATA.GGSRPT.EXTORD1` and so forth. When the file number reaches nine, the oldest file is deleted to make room for a new file, so there are never more than 11 files on the system at one time (the current report plus the ten aged reports).

To prevent the size of the report file from becoming too large, use the `REPORTROLLOVER` parameter in the Extract and Replicat parameter files. This parameter forces the report files to age on a regular schedule. Options are available to age the current file on a specific day and/or a specific time.

To minimize the impact of errors on the size of the Replicat report file, use the `WARNRATE` parameter in the Replicat parameter file. This parameter conserves the size of the report file and the event log by issuing a warning only after a specific number of errors have been generated, instead of after each one. This parameter is useful if you expect a certain number of errors and can tolerate them. The default for this parameter is to warn after 100 errors.

Generating Run-time Statistics

Run-time statistics show the current state of processing. By default, run-time statistics are written to the existing process report at the end of each run. To control when run-time statistics are generated, use the `REPORT` parameter. This parameter has options for controlling the day and time that statistics are generated.

To generate interim run-time statistics, use the `SEND EXTRACT` or `SEND REPLICAT` GGSCI command with the `REPORT` option syntax as shown below.

```
GGSCI> SEND {EXTRACT|REPLICAT} group_name
REPORT [time_option [RESET | FILE filename | TABLE name]]
```

The time_option controls the time span covered by the report, such as since the start of Extract or since the last report request. `RESET` sets the counters for that time_option to zero. `FILE` or `TABLE` limits the report to counts for *name*. For more information, see SEND REPORT.

To generate run-time statistics and also cause the report file to roll over to a new one, add the `ROLLREPORT` option to the command, for example:

```
GGSCI> SEND EXTRACT EXTORD, REPORT
GGSCI> SEND EXTRACT EXTORD, ROLLREPORT
```

Viewing Record Counts

You can produce record counts at designated times during processing by using the `REPORTCOUNT` parameter in the Extract and Replicat parameter files. Results are printed to the process report file and to screen.

The record count shows the number of records extracted and replicated since the Extract or Replicat process started. Counts can be obtained at regular intervals or each time a specific number of records is processed.

The STATS Command

To generate a statistical report for Extract or Replicat, specify the `LAGSTATS` parameter. Oracle GoldenGate measures lag in bytes and time:

- Lag in bytes is the difference between the position of the Extract program in the source at the time of the last checkpoint, and the current end-of-file. A lag value of `UNKNOWN` indicates that the process may have recently started and hasn't yet processed records, or that the source system's clock may be ahead of the target system's clock due to a reason other than time zone differences.
- Time lag reflects the lag in seconds at the time the last checkpoint was written. For example, if it is now 15:00:00, the last checkpoint was at 14:59:00 and the timestamp of the last record processed by the Replicat program was 14:58:00, the lag is reported as 00:01:00 (one minute, the difference between 14:58 and 14:59).

The report includes the following general performance categories:

- General statistics
- Lag statistics
- Extract's processing in the Oracle GoldenGate trail
- Audit trail reading statistics for Extract (when applicable)
- Output statistics for Extract only

The following table describes each item in the lag statistics report.

Item	Description
Last Record Timestamp	The timestamp of the source record (when the source record was input or committed to the target database).
Configured Interval	Determined by the `LAGSTATS INTERVAL` parameter.
Actual Duration	The duration of time measured.
Records Processed	Number of records output or replicated during the period.
Records per Second	Records processed per second during the interval.
Source Records per Second	The estimated rate of records read for either the TMF audit trails or logger processes.
Last lag	The time lag of the last record measured between (1) the records update into the source database and (2) the actual processing of the record by Extract or Replicat.
Min lag	Smallest value of Last lag during the interval.
Average lag	Average time lag calculated by adding the maximum lag during the interval to the minimum lag and dividing the result by 2.
Peak lag	Peak time lag during the interval for all records processed and the timestamp of the peak.
Last est. record lag	An estimate of the number of records the component is behind the current record.
Pct Below Lag of mi:ss:mmm	The percentage of times lag was below the time threshold you specified. This is an optional statistic, which can occur up to five times. To generate the statistic, specify the `THRESHOLD` option for `LAGSTATS`.
Pct CPU Busy	The amount of time the CPU in which Extract or Replicat was running was busy during the interval.

Item	Description
PCT Process Busy	The amount of time Extract or Replicat was busy during the interval.
At EOF?	Whether more data was available to process the last time more data was requested by Extract or Replicat from the audit or Oracle GoldenGate trails.
Trail Reads per Second	When reading Oracle GoldenGate trails, the number of attempted block reads per second.
Bytes per Trail Read	When reading Oracle GoldenGate trails, the number of bytes read per successful read.
Records per Block Read	When reading Oracle GoldenGate trails, the number of records read per successful read. This indicates the blocking factor on input.
Wait per Block Read	When reading Oracle GoldenGate trails, the amount of time Extract or Replicat waits, on average, to complete the read.
Audit Bytes per Second	The number of bytes of audit processed per second (TMF Extract only).
Pct EOF Trail Reads	For TMF Extract, the percentage of times Extract reached the end of file, compared with the number of records processed. For Replicat or Extract reading Oracle GoldenGate trails, the number of times the process read at the end of file, compared with the total number of blocks it attempted to read.
Transactions per Second	For TMF Extract, the number of transactions processed per second.
Transactions Aborted	For TMF Extract, the number of transactions aborted during the interval.
Audit positions	The number of times during the interval that Extract requested Audserv to position for read.
Audit position seconds	The elapsed time in seconds required for Audserv to position for read.
Audserv requests	The number of data requests to Audserv during the interval.
Audserv request wait seconds	The elapsed time for the Audserv to fulfill data requests during the interval.
Long transactions	The number of long transactions during the interval.
Long transaction seconds	The elapsed time for the long transactions that occurred during the interval.
Output Bytes per Second	For Extract, the bytes of data output to the extract trails per second.
Output Blocks per Second	For Extract, the number of blocks of data written to the Oracle GoldenGate trails per second.
Records per Block Written	For Extract, the average number of records in each block written.
Bytes per Block Written	For Extract, the average number of bytes in each block written.
Wait per Block Written	For Extract, the amount of time waiting for the last write or TCP/IP send to complete before sending the next block of data. This statistic can indicate whether the network might be introducing a delay to Extract.
Average Record Flush Delay	For Extract, the estimated average amount of time a record was held in its buffers before flushing.
Pct Output/Input	For Extract, the ratio of bytes output compared with input bytes processed. Useful for estimating potential bandwidth required by Extract based on amount of TMF audit generated.

Collecting Events from Other Systems

Event messages created by the Collector and Replicat on Windows and UNIX systems can be captured and sent back to EMS on NonStop systems. This feature enables centralized viewing of Oracle GoldenGate messages across platforms.

To collect events from other systems:

1. Run Collector on NonStop to collect and distribute EMS messages. For each EMSCLNT process, run one Collector process. The following example runs Collector and outputs its messages to `$0`.

   ```
   TACL> ASSIGN STDERR, $0
   TACL> RUN SERVER /NOWAIT/ -p 7880
   ```

2. Run the EMSCLNT utility on the remote target. EMSCLNT reads a designated error log and runs indefinitely, waiting for more messages to send. When EMSCLNT receives a message, it sends the message to a TCP/IP collector process on NonStop. See the examples for running EMSCLNT on other operating systems for syntax information.

Running EMSCLNT on Other Operating Systems

This Unix example reads the file `ggslog.err` for error messages. Error messages are sent to the collector to the NonStop at IP address `192.0.2.2` listening on port `7850`. The Collector on NonStop writes formatted messages to EMS Collector `$0`.

```
> $emsclnt -h 192.0.2.2 -p 7850 -f ggserr.log -c $0
```

This Windows example (from the DOS prompt) reads the file `d:\ggserrs\log.txt` for error messages. Error messages are sent to the Collector on host `ggs2` listening on port `9876`. The Collector on NonStop writes formatted messages to EMS Collector `$P0`.

```
> emsclnt -h ggs2 -p 9876 -f c:\ggs\ggserr.log -c $P0
```

Argument	Description
`-h ggs2`	The node on which the collector is being run. Can be a name or IP address. This is a required parameter.
`-p 9876`	The port at which the collector is listening for messages. This is a required parameter.
`-f c:\ggs\ggserr.log`	The error file from which EMSCLNT retrieves error messages. This is a required parameter.
`-c $P0`	The collector to which EMS messages should be written on the NonStop (default is $0).

10

Using Oracle GoldenGate Utilities

Oracle GoldenGate provides a number of utilities that support a variety of process requirements, such as generating data definitions and DDL.
This topic includes the following:

Generating Data Definitions with DEFGEN

When capturing, transforming, and delivering data across disparate systems and databases, you must understand both the source and target layouts. Understanding column names and data types is instrumental to the data synchronization functions of Oracle GoldenGate.

The DEFGEN utility produces a file defining the source files and tables' layouts. These definitions are used by the Collector and by Replicat. In some cases, Extract also uses a definition file containing the target layouts when transformation operations are required on the source system.

The output definitions are written and saved to a text file and transferred to all target systems in text format. When they start, Replicat and the Collector read the definitions to interpret the data formats read from Oracle GoldenGate trails.

> **Note**
>
> Do not modify the text file that is output from DEFGEN.

Once you have generated your definitions, you must specify their location in your process' parameter file. Replicat uses the SOURCEDEFS parameter to indicate which source definition file to use. Collector uses the -d argument at startup to specify which source definition file to use.

Run DEFGEN interactively or using a batch obey script.

```
TACL> RUN DEFGEN [/IN command_file/] [EXCLUDESYSTEM] [EXPANDDDL options]
[RECORDNAMEPROMPTING]
```

Option	Description
IN *command_file*	If you have created and saved a parameter file using the NonStop editor, enter the name of that file.
EXCLUDESYSTEM	Causes DEFGEN to omit the NonStop system name from the files and tables for which definitions are being generated.
EXPANDDDL *options*	Use the EXPANDDDL parameter to manipulate output for Enscribe record definitions containing arrays and redundant field names. This feature is primarily useful when mapping Enscribe files to SQL tables. It can also be useful when generating SQL tables based on Enscribe definitions using the DDLGEN utility. EXPANDDDL is not necessary when the source database is NonStop SQL.

Option	Description
RECORDNAMEPROMPTING	Use RECORDNAMEPROMPTING to enter the name of an existing record definition to use when generating a definition for a new table. Use this parameter to point to the same definition for multiple tables that have identical definitions made up of the same columns, column order, and data types.

For more information, see DEFGEN Arguments.

Configuring DEFGEN Interactively

1. Run DEFGEN from TACL using the following syntax:

   ```
   TACL> RUN DEFGEN EXPANDDDL EXPANDGROUPARRAYS RESOLVEDUPGROUP OMITREDEFS
   ```

2. In response to the prompts, enter information similar to the following example:

For the prompt:	Enter:
Enter definitions file name (or Exit):	$DATA1.GGSDEF.CUSTDEF
File/Table to create definition for (or Exit):	$DATA1.GGSSOU.ECUSTMER
Include DDL record definition (Y/N)?	Y
DDL dictionary:	$DATA1.GGSDDL
DDL record definition name:	ECUSTMER-REC
File/Table to create definition for (or Exit)	EXIT

3. Transfer this file, as a text file, to the target system.

Configuring DEFGEN in Batch

1. Use the NonStop editor to create a parameter file.
2. Enter parameters similar to the following examples:
   - For NonStop SQL

     ```
     $DATA1.GGSDEF.CUSTDEF
     $DATA1.GGSSOU.TCUSTMER
     EXIT
     ```

   - For NonStop Enscribe

     ```
     $DATA1.GGSDEF.CUSTDEF
     $DATA1.GGSSOU.ECUSTMER
     Y
     $DATA1.GGSDDL
     ECUSTMER-DEF
     EXIT
     EXIT
     ```

3. Start DEFGEN from TACL using a syntax similar to:

   ```
   TACL> RUN DEFGEN /IN GGSPARM.DEFGEN/
   ```

4. Transfer the generated definitions file, as a text file, to the target system.

A Sample Definitions File

```
Definition for table $DATA1.GGSSOU.TCUSTMER
Record length: 198
Syskey: 0
Columns: 13
TS          134 8 0 0 0 0 1 8  8 0 0 0 0 1 0 1
RECNUM      132 4 8 0 0 0 1 4  4 4 0 0 0 0 1 0 1
SYSNAME     1 8 12 0 0 0 0 8 8 8 0 0 0 0 1 0 0
TEXT        0 64 20 0 0 0 0 64 64 64 0 0 0 0 1 0 0
VAL1        134 8 84 0 0 0 1 8 8 8 0 0 0 0 1 0 0
VAL2        134 8  92 0 0 0 1 8 8 8 0 0 0 0 1 0 0
COL_COMPUTE 134 8 100 0 0 0 1 8 8 8 0 0 0 0 1 0 0
I16         130 2 108 0 0 0 1 2 2 2 0 0 0 0 1 0 0
I32         132 4 110 0 0 0 1 4 4 4 0 0 0 0 1 0 0
I64         134 8 114 0 0 0 1 8 8 8 0 0 0 0 1 0 0
I32_TOTAL   132 4 122 0 0 0 1 4  4 4 0 0 0 0 1 0 0
JTS         134 8 126 0 0 0 1 8 8 8 0 0 0 0 1 0 0
JTS_TEXT    0 64 134 0 0 0 0 64 64 64 0 0 0 0 1 0 0
End of definition
```

Running DEFGEN to Use Existing Definitions

Multiple tables that have the same structure (identical columns, column order, and data types) can use the same definition. To run DEFGEN for these tables, use the RECORDNAMEPROMPTING argument.

- Run DEFGEN from TACL using the following syntax:

  ```
  TACL> RUN DEFGEN RECORDNAMEPROMPTING
  ```

In response to the prompts, enter information similar to the following example:

For the prompt:	Enter:
Enter definitions file name (or Exit):	$DATA1.GGSDEF.CUSTDEF
File/Table to create definition for (or Exit):	$DATA1.GGSSOU.ECUSTMER
Use record name for definition file (Y/N)?	Y
Record or definition name to be used:	CUSTOMER-DEF
File/Table to create definition for (or Exit)	EXIT

Creating Target Database DDL

DDLGEN generates table definitions for target databases based on existing Enscribe and NonStop SQL definitions. It can also use the output from the DEFGEN utility (see above). DDLGEN

reduces the work necessary to create databases on platforms such as UNIX or Windows, and enables the creation of NonStop SQL databases based on Enscribe definitions. Target templates are provided for NonStop SQL, Oracle, SQL Server, DB2, and Sybase.

DDLGEN is run either interactively by supplying responses to user prompts, or in batch mode by supplying an input file.When running interactively, the user is supplied several prompts. For batch execution, the answers to the prompts are supplied in the obey file. It is recommended that the user runs the process in interactive mode to better understand the replies to specify in the obey file.

The result of running DDLGEN is a text file containing the create table statements. Transfer this file, as a text file, to your target system.

For more information, see DDLGEN Arguments in the Reference for Oracle GoldenGate on HP NonStop Guardian.

The DDLGEN syntax is:

```
TACL> RUN DDLGEN [/IN command_file/] [-d DEFGEN_output]
```

Argument	Description
IN *command_file*	If you have created and saved a file of responses using the NonStop editor, enter the name of that file. See the information on configuring DDLGEN interactively in Configuring DDLGEN Interactively for a list of the responses.
-d *DEFGEN_output*	Instructs DDLGEN to use the definitions file produced by DEFGEN.

Configuring DDLGEN Interactively

1. Start DDLGEN from TACL using a syntax similar to:

   ```
   TACL> RUN DDLGEN
   ```

2. In response to the prompts, enter information similar to the following example:

For the prompt:	Enter:
Output file for table DDL (or Exit):	$DATA1.GGSDEF.CUSTDEF
DDL template file name (or Exit):	*template_name* There are seven different templates prepackaged with Oracle GoldenGate: TMPLDB2 - DB2 for Windows and UNIX TMPLDB2M - DB2 for the z/OS and OS/390 TMPLMSA - Microsoft Access TMPLMSS - Microsoft SQL Server TMPLORA - Oracle TMPLSYB - Sybase TMPLTDM - NonStop SQL/MP
Source File/Table (or Exit):	$DATA1.GGSSOU.ECUSTMER

For the prompt:	Enter:
DDL dictionary:	`$DATA1.GGSDDL`
DDL record definition name:	`ECUSTMER-REC`
Source File/Table (or Exit):	`$DATA1.GGSSOU.TCUSTORD`
Source File/Table (or Exit):	`EXIT`

3. Transfer the definitions file, as a text file, to the target system.

Configuring DDLGEN in Batch

1. Use the NonStop editor to create a parameter file. For this example, the file name is `GGSPARM.DDLGEN`.
2. Enter parameters similar to the following examples:

```
$DATA1.GGSDEF.CUSTDEF
TMPLORA
$DATA1.GGSSOU.ECUSTMER
$DATA1.GGSDDL
ECUSTMER-REC
$DATA1.GGSSOU.TCUSTORD
EXIT
```

3. Initiate `DDLGEN`:

```
TACL> RUN DDLGEN /IN GGSPARM.DDLGEN/ -d $DATA1.GGSDEF.CUSTDEF
```

   In this step, you are instructing `DDLGEN` to use the `DEFGEN` definitions file that was produced by a previous run of `DEFGEN`.

4. Transfer the generated definitions file, as a text file, to the target system.

Addressing Enscribe DDL Peculiarities

Enscribe record definitions often contain the following items that do not map directly to SQL environments:

- `OCCURS` items

  SQL columns cannot have multiple occurrences, while Enscribe fields can.

- Group level items

  There is no grouping hierarchy in SQL, while an Enscribe record definition can contain fields that are redundant until qualified at the group level. This means that redundant column names can be created when mapping Enscribe definitions to SQL. For example, an Enscribe record might contain the field `YEAR` twice, once within the `BEGIN-DATE` group and once within the `END-DATE` group.

To get around these conditions, you can run the `DEFGEN` utility with various `EXPANDDDL` options set, as described in *Reference for Oracle GoldenGate on HP NonStop Guardian*. Use the `-d` parameter to specify the definitions file created by `DEFGEN` as input to `DDLGEN`.

Understanding the Template File

Templates are provided with each version of Oracle GoldenGate. A template file specifies how to generate the target definitions based on the source definitions. Each template file contains the following items:

- Literal text to output for each table definition.
- Source to target data type conversion specifications.
- Column name substitution specifications.
- Miscellaneous run-time parameters.
- Section headers.
- Comments, which begin with a pound sign (#).
- Session parameters, which are resolved at run-time by user prompts and applied during the entire `DDLGEN` session. Session parameters begin with a question mark (?).
- Per-table parameters, which are input by the user for each table definition generated. Per-table parameters begin with a percent sign (%).
- Calculated parameters, which include information determined by `DDLGEN`. Calculated parameters include the following:

Calculated Parameter	Description
`?TABLE`	The file name portion of the source table or file.
`?COLUMNS`	A list containing each column, its target data type, precision and scale (if any), and null/not null syntax.
`?KEYCOLUMNS`	A list containing each column in the primary key.

`?MAXPAGES`
`?MAXMEGS`
`?CURPAGES`
`?CURMEGS`
	The maximum and current number of 2048-byte pages and megabytes in the source table.

Sample Template File

This sample (`TMPLORA`), is a template file for converting HP NonStop DDL to Oracle DDL.

The sections are:

- Table creation section specifying operations for creating and managing tables. Note that in this section `?TABLE`, `?COLUMNS` and `?KEYCOLUMNS` are resolved by `DDLGEN`. `?TABLE_SPACE` is prompted for once and will apply to every table, while `%NEXT_SIZE` is prompted for on a per-table basis.
- Column name mapping section containing source NonStop and target Oracle column names. This section maps the source column names to the target names. In this example, any occurrences of `ROWID` in the NonStop database will be changed to `ROWID_` in the Oracle definition. If Oracle keywords appear in your NonStop database definitions, add entries to this list.
- Miscellaneous parameters section. Specify instructions for column formatting.

- Column type mapping section. Determines how NonStop types are defined in Oracle. `Precision` and `Scale` definitions can be `YES`, `NO` or a constant, positive value.

Note in this example, there are two entries for both `CHAR` and `VARCHAR`. Because Oracle allows a maximum of 255 characters in a `VARCHAR2`, we specify that all instances of `CHAR` and `VARCHAR` with length greater than 255 should become `LONGS`.

```
# Table Creation Section
#
DROP TABLE ?TABLE;
CREATE TABLE ?TABLE
(
?COLUMNS
,CONSTRAINT PK_?TABLE
PRIMARY KEY
(
?KEYCOLUMNS
)
USING INDEX
TABLESPACE ?TABLE_SPACE
)
TABLESPACE ?TABLE_SPACE
STORAGE (INITIAL 50K NEXT %NEXT_SIZE);
#
# Column Name Mapping Section
#
# NonStop column name     Oracle target name
#
ROWID                     ROWID_
SYSDATE                   SYSDATE_
#
# Miscellaneous Parameters Section
#
INCLUDENULL
#
# Column Type Mapping Section
#                                              Max   Max
#NonStop Type Target DB Type Precision  Scale Prec. Scale
#
CHAR          VARCHAR2       Y          N     255   N
CHAR          LONG           Y          N     N     N
VARCHAR       VARCHAR2       Y          N     255   N
VARCHAR       LONG           N          N     N     N
REAL          NUMBER         N          N     N     N
DOUBLE        NUMBER         N          N     N     N
NUMERIC       NUMBER         Y          Y     N     N
SMALLINT      NUMBER         Y          N     N     N
INTEGER       NUMBER         Y          N     N     N
LARGEINT      NUMBER         Y          N     N     N
DECIMAL       NUMBER         Y          Y     N     N
DATE          DATE           N          N     N     N
TIME          DATE           N          N     N     N
TIMESTAMP     DATE           N          N     N     N
DATETIME      VARCHAR2       Y          N     N     N
INTERVAL      VARCHAR2       Y          N     N     N
```

Sample NonStop SQL Table Definition

An example for creating Oracle table DDL is shown below.

```
CREATE TABLE TCUSTORD
(
```

```
    CUST_CODE        CHAR (4)                   NOT NULL,
    ORDER_DATE       DATETIME YEAR TO SECOND    NOT NULL,
    PRODUCT_CODE     CHAR (8)                   NOT NULL,
    ORDER_ID         NUMERIC (18)               NOT NULL,
    PRODUCT_PRICE    DECIMAL (8,2),
    PRODUCT_AMOUNT   DECIMAL (6,0),
    TRANSACTION_ID   NUMERIC (18)               NOT NULL,
    DESCRIPTION      CHAR (400),
    PRIMARY KEY (CUST_CODE, ORDER_DATE, PRODUCT_CODE, ORDER_ID)
);
```

Modifying the Sample Template File

In this example, you can make three modifications:

1. Add a `DATE_MODIFIED` column to each table and to the primary key.
2. Calculate the `NEXT` value based on the table's current size. Note that any parameter value that evaluates to a numeric value can be multiplied or divided.
3. Substitute the column name `ORDER_NUM` for instances of `ORDER_ID`.

The following is the template file, with modifications shown in bold.

```
#
# Table Creation Section
#
DROP TABLE ?TABLE;
CREATE TABLE ?TABLE
(
?COLUMNS
,  DATE_MODIFIEDDATE   NOT NULL
,  CONSTRAINT PK_?TABLE
   PRIMARY KEY
   (
?KEYCOLUMNS
,  DATE_MODIFIED
   )
   USING INDEX
   TABLESPACE ?TABLE_SPACE
)
TABLESPACE ?TABLE_SPACE
STORAGE (INITIAL 50K NEXT ?CURPAGES/5K);
#
# Column Name Mapping Section
#
# NonStop column name     Oracle target name
#
ROWID     ROWID_
SYSDATE   SYSDATE_
ORDER_ID  ORDER_NUM
```

Generating the Sample Definition

The following example generates an Oracle definition:

```
1> RUN DDLGEN
Output file for table DDL (or Exit): ORADDL
DDL template file name (or Exit): TMPLORA
Value for param TABLE_SPACE: USERS
Source File/Table (or Exit): $DATA1.SAMPLE.TCUSTORD
Source File/Table (or Exit): EXIT
```

The following is an example of the contents of ORADDL after DDLGEN is run.

```
DROP TABLE TCUSTORD;
CREATE TABLE TCUSTORD
(
   CUST_CODE       VARCHAR2(4)  NOT NULL
,  ORDER_DATE      DATE         NOT NULL
,  PRODUCT_CODE    VARCHAR2(8) NOT NULL
,  ORDER_NUM       NUMBER(18)  NOT NULL
,  PRODUCT_PRICE   NUMBER(8,2) NULL
,  PRODUCT_AMOUNT  NUMBER(6,0) NULL
,  TRANSACTION_ID  NUMBER(18)  NULL
,  DATE_MODIFIED   DATE        NOT NULL

,  CONSTRAINT PK_TCUSTORD
   PRIMARY KEY
   (
   CUST_CODE
,  ORDER_DATE
,  PRODUCT_CODE
,  ORDER_NUM
,  DATE_MODIFIED
   )
   USING INDEX
   TABLESPACE USERS
)
TABLESPACE USERS
STORAGE (INITIAL 50K NEXT 470K);
```

Using Syncfile

Syncfile manages non-database file duplication. For example, you may want to replicate configuration files that are small and change infrequently. This is a common requirement for maintaining a secondary system that has frequent database changes, but infrequent configuration file changes.

Syncfile can copy almost any type of file, making it suitable for other scenarios that require only infrequent, off-hours copying. By default, Syncfile uses the NonStop FUP DUP utility to perform file duplication; however, it can also run user-written TACL scripts to perform more specialized file duplication, such as FTP over TCP/IP.

You implement Syncfile by defining its parameters. The two main parameters include a file list to duplicate, and one to many schedules. A file set can be a file name, a wildcarded file name, or a file exclude list. The schedules are events which can be as frequent as you want, such as every day, every hour, or every ten minutes.

Syncfile options control the following:

- The files to duplicate
- The schedule for determining when files should be duplicated
- The method for duplication (FUP, FTP, etc.)
- Whether files should be always duplicated, or only when modified

A Syncfile parameter file can contain multiple schedules and file sets. In addition, you can create multiple Syncfile processes to support duplication for different applications or other requirements.

Syncfile processes are persistent. If a Syncfile process goes down unexpectedly, Manager automatically restarts it.

Implementing Syncfile

1. Create a parameter file with the necessary Syncfile parameters. The parameter file includes the names of the files to duplicate, the schedules, and other options.

   Sample Syncfile parameter file:

   ```
   EVENT DAILY, EVERY DAY AT 1:00, EXCLUDE FRIDAY, EXCLUDE AUGUST 2;
   EVENT FREQUENT, EVERY 2 HOURS;
   DUP $DATA1.SOURCE.*, TARGET \BKUP.$DATA2.*.*,
   ALWAYS, EVENT DAILY;
   DUP $DATA2.GGSPARM.*, TARGET \BKUP.$DATA3.*.*,
   TACLCMD "RUN $DATA1.GGSTACL.SYNCTCL<source> <target>",
   CHANGED, EVENT FREQUENT;
   ```

   This parameter file specifies the following attributes and actions:

   - Two events—daily and frequent. The daily event happens every day at 1:00 AM. However, the daily event is cancelled on all Fridays and August 2. The frequent event occurs every two hours. There are two `DUP` specifications. The first `DUP` specification indicates the file set `$data1.source.*`. Files satisfying that description are duplicated according to the daily event schedule (every day at 1:00). These files are duplicated regardless of whether the data has changed (the "always" option). Files are duplicated to `\bkup.$data2` with the same subvolume and file name as the corresponding source files.
   - By default, the `FUP DUP source, target, PURGE, SAVEALL` command is used to duplicate the files.
   - The second `DUP` specification names everything from `$data2.ggsparm` to be copied to `\bkup.$data3` with the same subvolume and file names. Files are duplicated on the frequent event schedule (every two hours). However, only those files with a modification timestamp on the source greater than that of the target will be duplicated (the changed option). The changed option does not have to be specified since it is the default option.

     In this example `TACLCMD` is added as a clause to the `DUP` parameter. This will cause Syncfile to run the TACL macro `$data1.ggstacl.synctcl` to duplicate the file. The macro is responsible for determining how to move source files to the target system, as well as any intermediate required steps.

     In this instance, the `<source>` and `<target>` arguments should not be replaced with any file name. They act as keywords to trigger Syncfile to use the `DUP $DATA2.GGSPARM.*, TARGET \BKUP.$DATA3.*.*` statement to identify the source and target parameters that will be passed to the macro.

     If file names are entered in `<source>` and `<target>` they will be passed to the macro instead. The following TACL macro and Syncfile parameters will pass `GGSPARM.FILE1` as %1% and `GGSPARM.FILEB` as %2% to cause it to duplicate `FILE1` to `FILEB`, not to `$DATA3.GGSPARM.FILEA`.

     ```
     ?TACLMACRO
     FUP DUP %1%, %2%
     DUP $DATA1.GGSPARM.FILE1, TARGET $DATA3.GGSPARM.FILEA,
     TACLCMD "RUN $DATA1.GGSPARM.TACL1 GGSPARM.FILE1 GGSPARM.FILEB",
     ALWAYS, EVENT DAILY 1330;
     ```

> **Note**
>
> Leaving out the <source> and <target> arguments will cause Syncfile to abend.

The two most important parameters for Syncfile are EVENT and DUP. At least one EVENT and one DUP parameter are required for each Syncfile operation. Each parameter entry must be terminated with a semi-colon (;).

2. Start GGSCI and add the Syncfile process.

```
TACL> RUN GGSCI
GGSCI> ADD SYNCFILE group_name
[, PARAMS parameter_file]
[, REPORT report_filename]
[, PROCESS process_name]
[, PROGRAM program_name]
```

   Add options as desired.

3. Start the Syncfile process.

```
GGSCI> START SYNCFILE
```

See GGSCI Commands for details about the GGSCI Syncfile commands and the Syncfile parameters.

A

Oracle Oracle GoldenGate Components

Here we discuss components of Oracle GoldenGate code used for customer implementations, such as programs, utilities, macros, libraries, and databases. There are also templates, demos, and sample code.
This appendix includes the following:

Programs, Utilities, Macros, and Libraries

Program/Macro	Description
AUDDUMP	A utility program to print the contents of the AUDCFG or GGSCPUxx segments.
AUDSERV	Reads audited database changes from TMF audit trails. Started by Extract processes. Must be owned by SUPER group and must have a PROG ID and license for other users to extract changed data from the audit trails.
BASELIB	The intercept library that is bound with programs to facilitate non-TMF based database change logging. The TNS mode can be used with programs written in TAL, and C.
BASELIBN	Native version of BASELIB that can be used with TAL, C, C++, and Java VM. On the operating systems, this library can be also be used with native COBOL programs.
BINDEXIT	TACL macro that merges Extract and Replicat with user exit routines.
BINDSKEL	TACL macro that merges BASELIB with the BASE24 SKELB library.
CHGNOTE	Notifies GGSLIB of configuration changes.
COMBLIB	A macro that combines the Oracle GoldenGate intercept library GGSLIB with a User Library that intercepts the same calls as Oracle GoldenGate.
COORDINATOR	Tracks the status of distributed network transactions to coordinate the processing across multiple nodes.
DBINIT	A macro that initializes the Oracle GoldenGate checkpoint and configuration files.
DDLGEN	Generates target database table definitions based on NonStop DDL.
DEFGEN	Generates source table definitions for Replicat to use for the translation of heterogeneous databases.
EMSDIST	Contains EMS messaging and tokens for Oracle GoldenGate messages.
EMSINST	A macro that installs EMS event message detail text.

Program/Macro	Description
ENTRYLIB	A macro that combines the Oracle GoldenGate intercept library BASELIBE with a User Library that intercepts the same calls as GGSLIB.
EVCXFUP	Contains the FUP scripts to create the EVENTCX file for custom EMS messages.
EXTRACT	Performs database change extract; formats and outputs results.
EXTRACTN	Native version of Extract.
EXTRR	Native relinkable version of Extract. Use when you have a native user exit to be linked into Extract. The NLDEXIT macro will reference this program.
FIXFLS	Adjusts alternate key pointers and turns auditing on proper Oracle GoldenGate files (for example, after duplicating from another location).
FUPLOG	Contains the FUP scripts to create the Logger configuration file LOGCONF.
GGSCI	User interface to Oracle GoldenGate functions.
GGSDLL	The default name of the executable version of native BASELIB for applications running on the operating systems. NLDLIB may combine BASELIB with other relinkables when creating GGDLL.
GGSLIB	User library bound with application programs to facilitate non-TMF based database change extract.
GGSLIBR	Native relinkable version of GGSLIB that can be combined with other code to create a larger combined user library.
GGSUNPAK	A macro that performs the UNPAK against Oracle GoldenGate installation files and runs the installation macro.
INSTALL	TACL macro that creates database files and performs other installation tasks.
KEYGEN	Program used to generate random keys. Used for encryption.
LEANBIND	A macro that removes the several routines out of BASELIB and creates LEANLIB. Use in place of BASELIB; however, you must contact Technical Support at Oracle GoldenGate before proceeding.
LOGDUMP	A program that provides the ability to display or search for information stored in log trails, extract trails, or extract flat files.
LOGDUMPA	A program that provides the same usage as LOGUDMP, but this must be used to access AES encrypted files.
LOGGER	Writes non-TMF audited database changes to log trails.
MGR	Carries out resource management functions as configured by system administrators.
MIGRATE	Moves checkpoint and other data from an old to a new Oracle GoldenGate installation.

Program/Macro	Description
NLDEXIT	A macro used to combine a native version of the intercept libraries.
NLDLIB	A macro used to create a native version of the intercept libraries.
REPLICAT	Replicates selected database changes from a set of source Enscribe files and SQL tables to a set of target files and tables.
REPR	A relinkable, native version of Replicat. Used when linking native user exits.
SCANGRP	A utility used by the MIGRATE utility to scan and convert group records.
SEGDUMP	A utility program used to examine the contents of the private context segment maintained by BASELIB. The context segment contains I/O buffers and state information for all files being logged.
SERVER	Another name for the Collector that receives data over TCP/IP and writes data to remote trails.
SFGEXIT	A program that can be activated within NonStop Safeguard to audit update access file opens and issue a message if no Oracle GoldenGate intercept program is bound in.
SQLCOMPS	List of SQLCOMP commands for Oracle GoldenGate programs.
SYNCFILE	Performs file replication of non-database files based on a user-set schedule.
TMFARLB2	A distributed TMF audit read library.
TMFARUL2	A distributed TMF audit read library.

Oracle GoldenGate Database

The Oracle GoldenGate database is created at installation time with the INSTALL macro. Each database file and its function are listed in the next table.

Database File	Description
AUDCFG	A dynamically created shared segment file used for GGSLIB configuration retrieval.
AUDSPEC	Contains audit management configuration parameters set up with GGSCI and read by Manager to perform audit management tasks.
CONTEXT	Contains Extract checkpoints to facilitate continuous processing of audit for particular Extract groups. Used by Manager to determine whether particular audit resources are still required.
EXTCTXT	Contains Extract checkpoints to track restart points within extract files in case Extract halts prematurely. Also contains information about individual extract trail dimensions and management.
EXTCTXT0	Alternate key file for EXTCTXT.

Database File	Description
GROUP	Contains each distinct Extract processing group.
LOGCONF	Stores logger configuration.
LOGGGS	Keeps log of critical events.
MRKRGGS	Audited file that accepts audit marker records.
REMCTXT	Contains the names of local and remote REPCTXT files to check before deleting extract files.
REPCTXT	Contains Replicat checkpoints to facilitate continuous processing of extract information. Used by Manager to determine whether particular extract file resources are still required.
REPCTXT0	Alternate key file for REPCTXT.
REPGRP	Contains each distinct Replicat processing group.
RMTCTXT	Contains checkpoints for remote extract files.
SYNCGRP	Contains each distinct Syncfile processing group.

External Component Summary

Component	Description
TMF Audit Trails	Contain change data information for source file and table insert, update and delete operations. Audit trails are read by Extract processes with Audserv for relevant database changes. Each audit trail can be read from one of four locations: the original location (on disk), from a disk dump, restored to disk from tape, or from a duplicate created by Manager.
Extract Files	Created by Extract processes. Extract files contain formatted database change records that can be input to Replicat processes, user-written applications or utilities. Extract processes can write to a single extract file, or a sequence of extract files known as extract trails. Manager purges extract files.
SQL and Enscribe Database Files	Database files that are the source of extract and target of delivery activities.
Parameter Files	Parameter files provide run-time parameters for all Oracle GoldenGate processes. There is no facility for entering parameters interactively, so parameter files are required.
Report Files	Extract and Replicat generate reports detailing statistical highlights of processing. Report files can be virtually any format, including spooler, edit files or the home terminal.
DDL Dictionaries	Dictionaries are an optional component of Extract and Replicat processing. DDL dictionaries describe Enscribe records processed by Extract and Replicat. This lets you describe record selection and column mapping criteria for both Extract and Replicat activities. The dictionaries must have been compiled using C30 or later DDL.

Component	Description
SQL Catalogs	Provide the definition of extracted SQL database change records. You need read access to the catalog associated with any source and target SQL tables. The catalogs allow Extract and Replicat to decode audit records and to build replicate SQL operations.
User Exit Routines	Object files you create that are bound in with Extract or Replicat to perform customized routines.
Error Console	Extract and Replicat send errors and warnings to both the error console and to the report file. The default error console is the home terminal. You can change the error console with the statement `PARAMS EMSLOG` *`EMS_collector`* before issuing the run command for either program. *filename* can include EMS distributors, processes, disk or spooler files.

Templates, Demonstrations, and Sample Code

File	Description
`DDLEXIT`	DDL for creating copy files for user exit and API routines.
`DEMOCL`	C include library for user exit demos.
`DEMOCOBL`	COBOL copy library for user exit demos.
`DEMOEDDL`	Sample DDL scripts for sample Enscribe files.
`DEMOFUPS`	Sample FUP scripts to create sample Enscribe source files.
`DEMOFUPT`	Sample FUP scripts to create sample Enscribe target files.
`DEMOOLDE`	Source code for the DEMOLEDO Enscribe demo program that performs inserts and updates for sample files.
`DEMOLDEO`	Program used to generate insert and update operations for sample Enscribe files.
`DEMOLDS`	Sample data for tables created by DEMOSQL.
`DEMOSQL`	SQL table creation DDL for sample TMF delivery User Tutorial example.
`DEMOXC`	An example user exit written in C.
`DEMOXCOB`	An example user exit written in COBOL.
`HELP`	Help file for GGSCI.
`TCPERRS`	Contains TCP/IP error handling parameters.
`TMPLMSA`	`DDLGEN` template file for Microsoft Access.

File	Description
TMPLMSS	DDLGEN template file for Microsoft SQL Server.
TMPLORA	DDLGEN template file for Oracle.
TMPLTDM	DDLGEN template file for Non Stop NS SQL.
TMPLSYB	DDLGEN template file for Sybase.
TMPLDB2	DDLGEN template file for DB2 on Windows and UNIX.
TMPLDB2M	DDLGEN template file for DB2 on z/OS and OS/390.
USEREXC	Blank template that can be used as a starting point for C user exits.
USEREXT	Blank template that can be used as a starting point for TAL user exits.
XLIBC	Include library for writing C user exits and interfacing to API.
XLIBCOB	Copy library for writing COBOL user exits and interfacing to API.

B

Installing Event Detail Text

Event Text makes Oracle GoldenGate detailed text available to operators using the Viewpoint event detail screen. Installing and working with Event Text is discussed here.

This appendix includes the following:

Standard Installation

To install event text (assuming a default configuration of Viewpoint), issue the following commands.

```
TACL> VOLUME installation_volume_subvolume
TACL> RUN EMSINST
```

The EMSINST macro performs the steps described in the following section.

Custom Installation

If your installation of Viewpoint differs from the default configuration, perform the following steps (Substitute appropriately for PATHMON process names, and so on.)

1. From TACL, enter the following:

   ```
   TACL> VOLUME GGS
   TACL> FUP /IN EVCXFUP/
   TACL> RUN EVCXLDO /IN EMSDTL/
   ```

2. Bring down the Viewpoint event detail server, as shown in these commands:

   ```
   TACL> PATHCOM $ZVPT
   =FREEZE SERVER ZVPT-EVNT-DETL
   =STOP SERVER ZVPT-EVNT-DETL
   =STOP SERVER ZVPT-EVNT-DETL
   ```

   If this is the first time that custom messages will be used on your system also perform the following assignment.

   ```
   =ALTER SERVER ZVPT-EVNT-DETL, ASSIGN CUSTOM-DETAIL, \SYS.$SYSTEM.SYSTEM.EVENTCX
   =EXIT
   ```

3. If $SYSTEM.SYSTEM.EVENTCX already exists, enter:

   ```
   TACL> FUP COPY EVENTCX, $SYSTEM.SYSTEM.EVENTCX, SHARE
   ```

   Otherwise, enter:

   ```
   TACL> FUP DUP EVENTCX, $SYSTEM.SYSTEM.EVENTCX
   ```

4. Bring up the Viewpoint event detail server, as follows:

   ```
   TACL> PATHCOM $ZVPT
   =THAW SERVER ZVPT-EVNT-DETL
   =START SERVER ZVPT-EVNT-DETL
   =EXIT
   ```

Customizing Error Messages

You can customize message text by altering the event text within the `EMSDTL` file before running the `EVCXLDO` or `EMSINST` programs.