Oracle® Fusion Middleware

Reference Guide for Oracle GoldenGate for HP NonStop (Guardian)

12*c* (12.3.0.1.0)
E99942-01
September 2018

Oracle Fusion Middleware Reference Guide for Oracle GoldenGate for HP NonStop (Guardian), 12*c* (12.3.0.1.0)

E99942-01

Contents

2 Oracle GoldenGate Parameters

5 DEFGEN Arguments

6 DDLGEN Arguments

7 User Exit Functions

Preface

This reference guide contains detailed information about parameters, commands, and functions for the HP NonStop Guardian platform.

Audience

This guide is intended for business analysts and system administrators who are planning, configuring, and running Oracle GoldenGate for HP NonStop (Guardian).

Documentation Accessibility

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

Accessible Access to Oracle Support

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

Related Information

The Oracle GoldenGate Product Documentation Libraries are found at

Oracle GoldenGate

Oracle GoldenGate Application Adapters

Oracle GoldenGate for Big Data

Oracle GoldenGate Plug-in for EMCC

Oracle GoldenGate Monitor

Oracle GoldenGate for HP NonStop (Guardian)

Oracle GoldenGate Veridata

Oracle GoldenGate Studio

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

Oracle GoldenGate A-Team Chronicles

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, such as "From the File menu, select **Save**." Boldface also is used for terms defined in text or in the glossary.
italic *italic*	Italic type indicates placeholder variables for which you supply particular values, such as in the parameter statement: `TABLE` *`table_name`*. Italic type also is used for book titles and emphasis.
`monospace` `MONOSPACE`	Monospace type indicates code components such as user exits and scripts; the names of files and database objects; URL paths; and input and output text that appears on the screen. Uppercase monospace type is generally used to represent the names of Oracle GoldenGate parameters, commands, user-configurable functions, 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
GGSCI Commands

Oracle GoldenGate Command Interface (GGSCI) allows the command-line interface between users and Oracle GoldenGate functional components. It gives access to users to edit, modify, update, synchronize, replicat, trail, database, and so, on. Here we describe the purpose and syntax for these commands.
The following table summarizes the functions that you can control with Oracle GoldenGate commands.

Command Group	Purpose
Manager commands	Start and stop the Manager program and determine whether it is running.
Extract commands	Establish Extract checkpoints and manage and monitor Extract processing.
Replicat commands	Establish Replicat checkpoints. Manage and monitor Replicat processing.
ER Commands	Allow you to manage Extract and Replicat groups as a unit with a single command. The commands you can use are the same `INFO`, `KILL`, `SEND`, `START`, `STAT`, and `STATUS` commands you would use for the Extract or Replicat.
Logger commands	Add and alter Logger configuration and manage Logger processes.
Trail commands	Create and manage Oracle GoldenGate trails.
Database commands	Supply information about data definitions and tables and encrypt logon password.
Audit trail commands	Determine audit trail management parameters and whether audit trail files are still required.
Remote checkpoint commands	Establish remote checkpoints that Manager checks before purging data used by Replicat processes.
TMF commands	Manage TMF dump information.
Coordinator commands	Start and stop the Coordinator program, manage and monitor Coordinator processing.
Process commands	Enable you to send commands to a process name instead of a group name.
Marker commands	Enable you to insert application-specific markers into audit trails or Logger trails to identify critical points in Extract and Replicat processing.
Programs commands	Bind `GGSLIB` intercept library into application programs for non-TMF audited database extraction.
Syncfile commands	Set up and manage Syncfile processes for duplicating files from one location to another.
Report commands	Allow you to scroll through Extract and Replicat processing reports.
Miscellaneous commands	Control various other aspects of Oracle GoldenGate.

Manager commands

The Manager module must be running for other Oracle GoldenGate components to operate. The process ensures proper startup, monitoring, and other activities. Once you start the Manager process, you can:

- Determine whether the Manager process is running
- Retrieve information about the running Manager process
- Stop the process

The Manager process (`$GGMGR`), runs as a NonStop process pair that includes the process (`$GGMGR`) and a child process (`$GGMGX`).

You can change the default process name from `$GGMGR` to another name. To change this and other default settings, see Changing Default Component Names.

INFO MANAGER

Use `INFO MANAGER` to determine whether the Manager process is running. If Manager is running, the process name, port number, TCPIP process, and IP address may be displayed depending on the parameters set for Manager.

Syntax

```
INFO MANAGER
```

Example

The following examples show different displays for the `INFO MANAGER` command.

Example 1
If the IP port is not configured, the display will be similar to:

```
Manager process $GGMGR is running.
(IP port not configured).
```

Example 2
If the TCPIP process name and port are configured, the display will be similar to:

```
Manager process $GGMGR is running
(Process \NY.$ZTC1, port 7830).
```

Example 3
If the IP port, TCPIP process name, and `IPINTERFACE` are all configured, the display will be similar to one of the following:

```
Manager process $GGMGR is running
(Process \NY.$ZTC1, IP 192.0.2.1 port 7830).

Manager process $GGMGR is running
(Process \NY.$ZTC1, IP 2001:db8:2010:5040:4fff:ffff:ffff:28 port 7830).
```

SEND MANAGER

Use `SEND MANAGER` to communicate with the current Manager process.

Syntax

```
SEND MANAGER
{CHILDSTATUS |
GETPORTINFO [DETAIL] |
GETPURGEOLDEXTRACTS |
KILL process_name}
```

CHILDSTATUS
Displays information about all processes started by Manager.

GETPORTINFO [DETAIL]
Retrieves the status of ports in use. Also returns statistical information about port sessions.
Include DETAIL with GETPORTINFO to retrieve information about all dynamically allocated ports, regardless of whether they are in use, as in:

```
SEND MANAGER, GETPORTINFO DETAIL
```

GETPURGEOLDEXTRACTS
Displays information about trail maintenance rules set with the PURGEOLDEXTRACTS parameter in the Manager parameter file. For more information about PURGEOLDEXTRACTS, see "PRIORITY".

KILL *process_name*
Stops a process that was previously created by Manager. Manager returns an error if the process is not one it created.

Examples

Example 1

SEND MANAGER CHILDSTATUS returns a child process status similar to the following.

```
ID  Process         Retry   Retry Time         When added
--- --------------- ------- ----- ------------ ------------------
  3 \NY.$GGS03      0,0512      0 None         2014/08/16 14:39:07
```

Example 2

SEND MANAGER GETPURGEOLDEXTRACTS returns a report similar to the following:

```
PurgeOldExtracts Rules
Fileset                       MinHours  MinFiles     UseCP
$DATA04.GGSDAT.HT*                   0         1       Y
$DATA04.GGSDAT.ET*                   0         1       Y

Extract Trails
Filename                    Group    Oldest Seqno MinHours
\NY.$DATA04.GGSDAT.ET       REPACL              0        0
\NY.$DATA04.GGSDAT.HT       ACTHIST             0        0
\NY.$DATA04.GGSDAT.LT       ACRLOG              0        0
\NY.$DATA04.GGSLOG.LT       ACELOG              0        0
```

START MANAGER

Use START MANAGER to start the Manager process.

Syntax

```
START MANAGER
[, CPU primary_cpu]
[, BACKUPCPU backup_cpu]
[, PRI priority]
```

CPU *primary_cpu*
The primary CPU name.

BACKUPCPU *backup_cpu*
The backup CPU name. If a backup CPU is specified in the Manager parameter file, it overrides any `START MANAGER` backup CPU specification.

PRI *priority*
Sets the NonStop priority of the process.

Example

```
START MANAGER, CPU 1, BACKUPCPU 3, PRI 170
```

STATUS MANAGER

Use `STATUS MANAGER` to determine whether the Manager process is running and to identify its characteristics.

Syntax

```
STATUS MANAGER
```

Example

The command `STATUS MANAGER` will display the Manager process name, the running process, and the port number as shown below.

```
Manager process $ACMGR is running (IP \NY.$ZTC1 port 7670)
```

If a specific IP address or DNS name has been assigned using `IPINTERFACE`, it will be included as shown below.

```
Manager process $ACMGR is running (IP \NY.$ZTC1 192.0.2.2 port 7670)
```

STOP MANAGER

Use `STOP MANAGER` to stop the Manager process. You will be asked to confirm this command, since stopping Manager eliminates important activities. GGSCI logs `STOP MANAGER` commands to the Oracle GoldenGate event log.

Syntax

```
STOP MANAGER [!]
```

!
Unless you specify the exclamation point (!), you must confirm this operation.

Extract commands

Use Extract commands to create and manage Extract groups. The Extract process captures operations and sends the data to the target system. The Extract process maintains checkpoints to provide a starting point for subsequent runs, provides run history information, and displays the audit trails required for a given Extract group.

Process names, parameter files, and report files take system-assigned default values. Oracle GoldenGate Software recommends using the default names. If your installation requires different names see Changing Default Component Names.

ADD EXTRACT

Use `ADD EXTRACT` to add Extract groups, allowing change records to be processed from run to run without data loss.

Using `ADD EXTRACT` options you can perform the operations that are summarized in "ADD EXTRACT options summary".

Syntax

```
ADD EXTRACT group_name
{
[, BEGIN time |, AUDSEQNO seq_num, AUDRBA rba] |
[[, SOURCE trail_name
  {BEGIN time |, EXTSEQNO seq_num, EXTRBA rba}] |
[, LOGTRAILSOURCE trail_name
  {BEGIN time |, EXTSEQNO seq_num, EXTRBA rba}] |
[, SOURCEISTABLE]] |
[, FILETYPE file_type file_name]
}
[, CPU primary_cpu]
[, BACKUPCPU backup_cpu]
[, PRI priority]
[, PROCESS process_name]
[, PROGRAM program_name]
[, PARAMS param_file_name]
[, REPORT report_name]
[, DESC "text"]
```

ADD EXTRACT options summary

group_name
The group name.

SOURCE *trail_name* | LOGTRAILSOURCE *trail_name* | SOURCEISTABLE | FILETYPE *file_type, file_name*
The default source for `ADD EXTRACT` is the `TMF` audit trail. For information on other data sources see "Specifying the Data Source".

BEGIN *time* | , {AUDSEQNO *seq_num* , AUDRBA *rba* | EXTSEQNO *seq_num*, EXTRBA *rba*}
To specify a begin time or starting point in an audit trail or an Oracle GoldenGate trail, see "Specifying a Starting Point".

CPU *cpu* BACKUPCPU *cpu* PRI *priority*
To specify the CPUs, see "Assigning CPUs".

DESC "text"
See "Describing the Group".

PARAMS *param_file_name* REPORT *report_name*
See "Specifying an Alternative Parameter or Report File".

PROCESS *process_name*
See "Specifying an Alternative Process".

PROGRAM *program_name*
The name of the object file to run. See "Executing user exits".

Example

The following example creates an Extract group called `DISTRIB` that:

- Begins at midnight on May 1, 2010
- Runs in CPU `9` at priority `170` with an assigned backup CPU in case the primary fails

```
ADD EXTRACT DISTRIB, BEGIN 2010-05-01 00:00, CPU 9, BACKUPCPU 7, PRI 170
```

Specifying the Data Source

The default `ADD EXTRACT` source is a TMF audit trail. If your source is not the audit trail, you can specify an alternative source. Valid sources are:

- A local Oracle GoldenGate trail
- An Oracle GoldenGate Logger trail
- An entry-sequenced or BASE24 `TFL/PTLF` file
- Data captured directly from a file or table for one-time processes, such as initial synchronization

Using a Local Oracle GoldenGate Trail

A local Oracle GoldenGate trail is specified by `SOURCE` *`trail_name`*. The following example identifies the data source as a local Oracle GoldenGate trail, and specifies a sequence number in the trail at which to begin extracting data.

```
ADD EXTRACT FINANCE, SOURCE \LA.$D1.GGSDAT.AA, EXTSEQNO 26
```

Not Creating the : Use the `NOCREATE` option of `SOURCE` to specify that the trail is not created. If the `CREATE` option or no value is specified, the trail is created.

Using the Logger Trail

A Logger trail is specified by `LOGTRAILSOURCE` *`trail_name`*, as in:

```
ADD EXTRACT FINANCE, LOGTRAILSOURCE $DATA2.GLOGGGL.AA
```

Using a File

An entry-sequenced or ACI file source is specified by `FILETYPE` *`file_type file_name`*, as in:

```
ADD EXTRACT DISTRIB, FILETYPE ENTRY $DATA5.GGSDAT.FL1234
```

- For *file_name*, enter one of: ENTRY, ACITLF, ACIPTLF, ACITLFX, ACIPTLFX, or ADVANTAGE.
- Include the ALTINPUT and RANGE parameters in the Extract parameter file when capturing directly from a sequence of files

F

For One-time Processing

Initial synchronization or other one-time tasks are specified by SOURCEISTABLE (or SOURCEISFILE for an Enscribe file), as in:

```
ADD EXTRACT GROUP1, SOURCEISTABLE
```

When you configure Extract for a task, you must include a corresponding SOURCEISTABLE parameter in the Extract parameter file.

SOURCEISTABLE does not maintain checkpoints unless RESTARTCHECKPOINTS is used.

Specifying a Starting Point

You can specify a trail file sequence number and relative byte address as a starting point within an audit trail or local Oracle GoldenGate trail. However, it is more typical to specify a starting point using BEGIN with a date and time, which is the preferred method.

BEGIN *time*
Determines when Extract begins processing data in the audit trail. The *time* options are: NOW, or a date and time as yyyy-mm-dd [hh:mi:[ss[.cccccc]]].

AUDSEQNO *seq_num*
Identifies the TMF audit trail file sequence number at which to begin extracting data

AUDRBA *rba*
Specifies that processing begin at the specified relative byte address.

EXTSEQNO *seq_num*
Identifies the Oracle GoldenGate trail file sequence number at which to begin extracting data.

EXTRBA *rba*
Specifies that processing begin at the specified relative byte address.

Example

```
ADD EXTRACT ORDERS, BEGIN NOW
```

Assigning CPUs

When you add an Extract group you can specify primary and backup CPUs and a process priority.

CPU *cpu*
The primary CPU in which Extract runs. The default is the CPU in which Manager runs.

BACKUPCPU *cpu*
An alternative CPU on which Extract runs if the primary CPU becomes unavailable.

PRI *priority*
The NonStop priority for the process. This defaults to the NonStop priority assigned to the TACL process underlying the ADD.

Example

This example assigns both the primary and backup CPUs and a priority.

```
ADD EXTRACT DISTRIB, BEGIN 2010-05-01 00:00, LOGTRAILSOURCE $DATA2.GLOGGGL.AA, CPU
9, BACKUPCPU 7, PRI 170
```

Specifying an Alternative Process

The default process name is $GGSnn, where nn represents the sequence of the process. Oracle GoldenGate recommends that you use the default, however, if you must specify an alternative process, you can do so with the PROCESS *process_name* option.

Example

```
 ADD EXTRACT FINANCE, BEGIN 2010-05-01 00:00, PROCESS $GGE07
```

Specifying an Alternative Parameter or Report File

Oracle GoldenGate recommends that you use the default parameter and report names, however, if you must specify an alternative name, use the options described here. Alternatively, you can change the default names globally from the GLOBALS parameter file using ADD DEFINE. See the parameter summary for GLOBALS on "GLOBALS Parameters Summary". Also see Changing Default Component Names.

- The default parameter file name is *GGS_volume*.GGSPARM.*group_name*, where *group_name* represents a group, such as FINANCE.
- The default report file name is *GGS_volume*.GGSRPT.*rpt_name*, where *rpt_name* represents the group name, such asFINANCEOracle GoldenGate creates an entry-sequenced file to hold each group's run results, and by default, the report name is the same as the group name.

To change the default names:

PARAMS *param_file_name*
Supplies an alternative parameter file name. Enter the fully qualified path name for the parameter file.

REPORT *report_name*
Supplies an alternative report file name. Enter the fully qualified path name for the parameter file.

Example

These examples change the default parameter file and report names.

```
ADD EXTRACT FINANCE, BEGIN 2010-05-01 00:00, PARAMS $DATA01.NEWPARM.FINANCE
ADD EXTRACT FINANCE, BEGIN 2010-05-01 00:00, REPORT $PROD.NEWRPT.FINANCE
```

Describing the Group

Use the DESC "*text*" option to describe an Extract group.

Example

```
ADD EXTRACT ET24AT2, LOGTRAILSOURCE GGSLOG.LT,
DESC "T24 data pump for ATM transactions to IBM in Seattle"
```

Executing user exits

You can create and run your own routines by compiling them into an object file and binding this to the Extract program using the TACL macro named `BINDEXIT`. For more information, see Creating User Exits.

When you are ready to call the user exit, launch the Extract object that has the bound routines with the `PROGRAM` *`program_name`* option. Manager uses that program when starting the process.

Example

```
ADD EXTRACT GROUP1, BEGIN NOW, CPU 1, PRI 150, PROGRAM $DATA.GGS.FINEXIT1
```

ALTER EXTRACT

Use `ALTER EXTRACT` primarily to change attributes of the `CPU`, `PRIORITY` or `BACKUPCPU` options. You can use `ALTER EXTRACT` to change attributes of the options you specified with `ADD EXTRACT`, but you should consider the following:

- Use caution when changing the `BEGIN` values previously set with `ADD EXTRACT`. Since the `BEGIN` option checkpoints the starting point in the source, changing it may cause duplicate or missing records.
- You can change `EXTRAILSOURCE` or `LOGTRAILSOURCE` settings with `ALTER EXTRACT`, but Oracle GoldenGate recommends deleting and re-adding the group instead.

Syntax

```
ALTER EXTRACT group_name
[, ETROLLOVER]
[, ETPURGE]
[, option ]
```

`group_name`
The group name.

`ETROLLOVER`
Causes Extract to increment and write to the next file in the trail sequence when restarting. For example, if the current file is `ET000002`, the current file will be `ET000003` when `Extract` restarts.

`ETPURGE`
Causes old trails to be purged before the new one is created. Valid only when `ETROLLOVER` is specified.

`option`
In addition to the above described options, you can specify any appropriate `ADD EXTRACT` option.

CLEANUP EXTRACT

Use `CLEANUP EXTRACT` to delete old run history records for a group. This command keeps the last run record, enabling processing to resume from the correct position.

For example: `CLEANUP EXTRACT FINANCE` deletes the run history records for the `FINANCE` group, and keeps the last run record. You can also specify a quantity of records to save, as in: `CLEANUP EXTRACT * SAVE 5`, saving the last five run records.

Syntax

```
CLEANUP EXTRACT group_name [, SAVE count ]
```

group_name
An Extract group name or wildcard specification, such as `*` or `FIN*`.

SAVE *count*
Save the last `option` run records instead of just the last record.

DELETE EXTRACT

Use `DELETE EXTRACT` to delete an Extract group and its associated checkpoints. Use this when the TMF configuration changes, or when you no longer require the group.

When you delete an Extract group, Oracle GoldenGate deletes both the group and the metadata that controls the group's trail. By default it retains all the files currently in the trail. If you want to delete the trail files, you must use the exclamation point (!) in the `DELETE EXTRACT` statement or manually purge the files.

Syntax

```
DELETE EXTRACT group_name [!]
```

group_name
An Extract group name or wildcard specification, such as `*` or `FIN*`.

!
(exclamation point) Deletes trail files associated with each group without prompting the operator.

INFO EXTRACT

Use `INFO EXTRACT` to retrieve processing history for an Extract group. You can specify reporting options to obtain:

- Status of the process
- The process run history
- A process lag report
- Detailed historical checkpoints
- Only processes that are running, or stopped
- Information about tasks

Syntax

```
INFO EXTRACT group_name
[, BRIEF | DETAIL]
[, LAG number SECONDS | MINUTES | HOURS]
[, SHOWCH]
[, UP | DOWN]
[, TASKS | ALLPROCESSES]
[, PROGRAM]
```

group_name
An Extract group name or wildcard specification, such as `*` or `FIN*`.

BRIEF
Reports:

- Status of the process (`STARTING`, `RUNNING`, `STOPPED` or `ABENDED`).
- An approximation of the time and byte lag between the associated source and Extract processing.

DETAIL
Reports:

- Process run history, which includes starting and stopping points within the audit.
- Run history for trails.
- Process parameters established by the `ADD EXTRACT` command.

LAG *number* SECONDS | MINUTES | HOURS
Restricts the display to groups that are a specified time interval behind. This helps spot critical conditions. The lag returned by this command is approximate. For precise information, use `LAG EXTRACT`. Lag measures both bytes behind and time behind. For more information about how Oracle GoldenGate reports lag, see Changing Default Component Names.

SHOWCH
Shows detailed historical checkpoints.

UP | DOWN
Shows processes that are either running, (`UP`) or not (`DOWN`). Specify either `UP` or `DOWN`.

TASKS | ALLPROCESSES
Shows information about either tasks or all processes that are running. Specify either `TASKS` or `ALLPROCESSES`.

PROGRAM
Displays the name and location of the object that is running.

KILL EXTRACT

Use `KILL EXTRACT` to force Extract to stop immediately. Try `STOP EXTRACT` first because it also performs cleanup. Using the Oracle GoldenGate commands `STOP` or `KILL` is preferred to stopping processes from TACL. Manager automatically restarts processes that are stopped from TACL.

Syntax

```
KILL EXTRACT group_name
```

group_name
The group name. You can use wildcards to kill a set of groups.

LAG EXTRACT

Use `LAG EXTRACT` to determine Extract's relative position in the audit trail. This command estimates the lag behind the source database more precisely than `INFO EXTRACT`.

For more information about how Oracle GoldenGate reports lag, see Changing Default Component Names.

To determine lag for local processes, specify the group name. To determine lag for remote processes, specify the remote process name.

Syntax

```
LAG EXTRACT {group_name | process_name}
```

group_name
The group name, as in: `LAG EXTRACT FINANCE`

process_name
The process name, as in: `LAG EXTRACT $GGE00`

SEND EXTRACT

Use `SEND EXTRACT` to communicate with a running Extract process. Using `SEND EXTRACT` options, you can perform a variety of operations that are summarized in "SEND EXTRACT options summary".

Syntax

```
SEND EXTRACT group_name {
ARCLOSECATALOG |
AUDITEND |
STATUS |
GETTCPSTATS |
RESETTCPSTATS |
REPORT [time_option [RESET | FILE name | TABLE name]] |
ROLLREPORT |
GETEXTARSTATS |
RESETEXTARSTATS |
GETARSTATS, [MAT | AUXnn] |
RESETARSTATS, [MAT | AUXnn] |
GETTRANSINFO |
GETARPROCESS |
GETARPARAMS, [MAT | AUXnn] |
GETARFILELIST, [MAT | AUXnn]]
GETARFILESTATS, [FILE | MAT | MINRECS | RESET | QUIET |
NOPARTITIONS] |
GETAREXCLUDELIST, [FILE | MAT | AUXnn] |
CLEAREXCLUDELIST |
ROLLOVER |
```

```
LAGSTATS option |
LAGSNAPSHOT |
LAGREPORTON |
LAGREPORTOFF |
LAGOFF |
FORCESTOP |
STOP |
GETROLLBACKS |
IGNOREROLLBACKS
}
```

group_name
A running Extract group. If the group specified is not running, an error is returned.

SEND EXTRACT options summary

AUDITEND | STATUS | REPORT | GETTCPSTATS | RESETTCPSTATS
See "Obtaining process reports".

ARCLOSECATALOG | GETEXTARSTATS | RESETEXTARSTATS | GETARSTATS | RESETARSTATS | GETTRANSINFO | GETARPROCESS | GETARPARAMS | GETARFILELIST | GETARFILESTATS | GETAREXCLUDELIST | CLEAREXCLUDELIST
See "Managing the Audserv program".

ROLLREPORT
See "Opening a new report file".

ROLLOVER
See "Rollover Oracle GoldenGate trails".

LAGSTATS *option*
See "Obtaining lag reports".

FORCESTOP | STOP
See "Stopping the process".

GETROLLBACKS | IGNOREROLLBACKS
See "Processing rollbacks".

Example

```
SEND EXTRACT FINANCE, STOP
SEND EXTRACT MANUFACT, ROLLOVER
```

Obtaining process reports

You can generate reports for:

Report	Option	Description
End of audit trail	AUDITEND	Queries the Extract process to determine whether all records in the audit trails have been processed. This command indicates whether more Extract and 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.

Report	Option	Description
Processing status	STATUS	Returns a detailed status of the processing state, including current position and activity.
Processing statistics	REPORT	Generates an interim statistical report to the report file, including the number of inserts, updates, and deletes. Refer to "SEND REPORT" for detail on SEND REPORT options.
TCP/IP statistics	GETTCPSTATS	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.
TCP/IP statistics type	RESETTCPSTATS	Resets the TCP/IP statistics so the next report displays fresh statistics.

Example

The first example uses the AUDITEND option to report on the end of an audit trail. The second example specifies the STATUS option to return details of the processing state.

```
SEND EXTRACT FINANCE, AUDITEND
SEND EXTRACT FINANCE, STATUS
```

Opening a new report file

To close the current report file and open a new one, specify the ROLLREPORT option. ROLLREPORT renames the current file by appending a number to the end of the report name (such as EXTACCT0), then opens a new report file with the original name.

Managing the Audserv program

SEND EXTRACT supplies the following options for determining the status of Audserv operations.

ARCLOSECATALOG
Instructs Audserv to close its opens on the SQL Catalog.

GETEXTARSTATS
Retrieves information about Audserv activity. Information returned includes: first and last record timestamp, first and last read timestamp, bytes processed, commits, and other processing statistics.

RESETEXTARSTATS
Resets the report generated by GETEXTARSTATS.

GETARSTATS, [MAT | AUXnn]
Retrieves audit trail statistics from Audserv.

RESETARSTATS, [MAT | AUXnn]
Resets the report generated by GETARSTATS.

GETTRANSINFO
Retrieves information from Extract's pending transaction table.

GETARPROCESS
Retrieves the process names of Audserv processes.

GETARPARAMS, [MAT | AUXnn]
Retrieves Audserv run-time parameters.

GETARFILELIST, [MAT | AUXnn]
Retrieves the Audserv file list.

GETARFILESTATS, [FILE | MAT | MINRECS | RESET | QUIET | NOPARTITIONS]
Retrieves Audserv file level statistics.

GETAREXCLUDELIST, [FILE | MAT | AUXnn]
Retrieves the contents of the Audserv exclude list.

CLEAREXCLUDELIST
Clears the Audserv exclude list.

Rollover Oracle GoldenGate trails

The `ROLLOVER` option closes the current trail and opens the next trail in the sequence.

Obtaining lag reports

`SEND EXTRACT` supplies options for generating a variety of lag reports.

LAGSTATS *option*
Retrieves and optionally reports lag statistics. The options are the same as those for the `LAGSTATS` parameter. See additional `LAGSTATS` information "LAGSTATS".
The `SEND EXTRACT LAGSTATS` specification replaces any previous `LAGSTATS` entry.

LAGSNAPSHOT
Writes a current statistics report to the screen and to the report file. To generate this report, specify either the `LAGSTATS` parameter in the parameter file, or issue `SEND EXTRACT group_name, option`.

LAGREPORTON
Generates a report for each lag interval.

LAGREPORTOFF
Turns off automatic reporting, but continues to retrieve data.

LAGOFF
Turns off lag statistics.

Stopping the process

You can stop the current process with:

FORCESTOP
Terminates the process with a `STOP` operation.

STOP
Terminates the run gracefully. This command is preferable to stopping from TACL, which results in an `ABEND` status.

Processing rollbacks

Process rollback records with:

GETROLLBACKS
Retrieves rollback records. Use this command only before extracting changes during an initial-load phase.

IGNOREROLLBACKS
Ignores rollback records. Use this command after completing your initial load.

START EXTRACT

Use `START EXTRACT` to start Extract. GGSCI routes the `START` request to Manager to start and monitor the process.

Syntax

```
START EXTRACT group_name
```

group_name
The name of the Extract group. You can use wildcards to specify a set of group names, such as, `*` or `*FIN*`.

STATUS EXTRACT

Use `STATUS EXTRACT` to determine if Extract groups are running. A report displays to the Extract process's home terminal.

Syntax

```
STATUS EXTRACT group_name
[, DETAIL] | [,TASKS | ALLPROCESSES]
```

group_name
The name of the group. You can use wildcards to specify a set of group names, such as, `*` or `*FIN*`.

DETAIL
When you specify `DETAIL`, (`STATUS EXTRACT *, DETAIL`) the audit trails required by the group are also listed. Output consists of the locations of required audit trails, whether they are on disk or tape, and whether the trails still exist.
`DETAIL` is useful for determining whether audit must be restored from tape before the group is run and which groups are causing Manager to tie up TMF resources.

TASKS | ALLPROCESSES
Determine either the tasks or all processes that are running. Specify either `TASKS` or `ALLPROCESSES`.

STOP EXTRACT

Use `STOP EXTRACT` to stop Extract gracefully. Use `STOP` when you are changing the process configuration and to prevent Manager from automatically restarting the process.

Syntax

```
STOP EXTRACT group_name [, WAIT [seconds] | ATEND |!]
```

group_name
The name of the Extract group. You can use wildcards to specify a set of group names, such as, * or *FIN*.

WAIT *seconds*
GGSCI waits for Extract to terminate before issuing the next prompt. If *seconds* is specified, GGSCI waits that many seconds before returning control to the user. If you do not specify WAIT, GGSCI issues the next prompt immediately.

ATEND
Instructs Extract to stop when it reaches end-of-file for the last sequence of audit trails. If the application that updates the source database is brought down first, this ensures that Extract processed all relevant database updates before stopping.
If Extract is reading data from an Oracle GoldenGate trail instead of TMF audit trails, ATEND causes Extract to terminate when end-of-file is reached for the last sequence of the trails.

!
(Exclamation point) Stops Extract immediately, even in the middle of a transaction. Use this option to terminate long running transactions. As with ATEND, a grouped transaction is rolled back but the individual transactions are replayed, if the trail is available.

Replicat commands

With Replicat commands, you can establish initial checkpoints so that data can be continuously and accurately processed. After the initial run, these checkpoints provide a starting point for subsequent runs. Replicat commands also provide run history information.

Replicat process names, parameter files and report files take system assigned default values. To change these default settings, see Changing Default Component Names.

ADD REPLICAT

Use ADD REPLICAT to add a Replicat group. A Replicat group allows data changes to be processed from run to run without missing records.

Using ADD REPLICAT options you can perform a variety of operations that are summarized in the argument table.

Syntax

```
ADD REPLICAT group_name
{, SPECIALRUN |  trail_name | LOGTRAIL trail_name}
[, BEGIN time |, EXTSEQNO seq_number, EXTRBA rba]
[, CPU primary_cpu]
[, BACKUPCPU cpu]
[, PRI priority]
[, PROCESS process_name]
[, PARAMS param_file_name]
[, REPORT report_name]
```

```
[, DESC "text"]
[, PROGRAM program_name]
```

group_name
Required. Up to 7 characters to designate some logical function of this Replicat group. You can use wildcards to specify a set of group names, such as, `*` or `*FIN*`.
To use a group name of up to 10 characters, you can use the global parameter `OLDGROUPNAMING`. However, Oracle GoldenGate recommends constraining group names to 7 characters.

SPECIALRUN | *trail_name* | LOGTRAIL *trail_name*
Either `SPECIALRUN` or one of the two record sources is required. A warning is issued if the specified record source does not exist.

- For `SPECIALRUN` see "Configuring initial data synchronization or other tasks".
- Specify *`trail_name`* when the source is a local trail.
- Specify `LOGTRAIL` *`trail_name`* where the record source is a Logger trail.

BEGIN *time* EXTSEQNO *seq_num*, EXTRBA *rba*
See "Specifying a starting point".

CPU *primary_cpu* BACKUPCPU *cpu* PRI *priority*
See "Assigning CPUs".

DESC "*text*"
See "Enter a Replicat group description".

PARAMS *param_file_name* REPORT *report_name*
See "Specifying an alternative parameter or report file".

PROCESS *process_name*
See "Specifying an alternative Replicat process".

PROGRAM *program_name*
The object to be run. See "Executing user exits".

Example

This Replicat group reads data from a trail created and stored at `\NY`. It starts processing from the beginning of the `AA` trail and runs on `CPU 5` at a priority of `160`.

```
ADD REPLICAT FINANCE,  \NY.$DATA2.EXTDAT.AA,
CPU 5, PRIORITY 160
```

Specifying a starting point

Normally, Replicat begins processing at the beginning of the trail. However, you can control when and where Replicat begins processing with one of the following options:

BEGIN *time*
Determines when Replicat begins processing data the audit trail. The *`time`* options are:

- `NOW`
- A date/time in the format `yyyy-mm-dd [hh:mi:[ss[.cccccc]]]`

Note: Using `BEGIN` is not recommended, because it causes Replicat to bypass data preceding the specified begin point and can cause the target data to be out of synchronization.

`EXTSEQNO` *`seq_number`*
Identifies a specific sequence number in the local Oracle GoldenGate trail at which to begin extracting data.
For example, if the is `$SYSTEM.GGSDAT.ET` and `EXTSEQNO` is `26`, processing begins in trail file `$SYSTEM.GGSDAT.ET000026`. Omit this parameter unless special circumstances arise.

`EXTRBA` *`rba`*
Specifies that processing begin in the local Oracle GoldenGate trail at the specified relative byte address.

Example

Example 1

```
ADD REPLICAT ORDERS,  \NY.$DATA2.EXTDAT.AA, BEGIN NOW, CPU 6, PRI 170
```

Example 2

```
ADD REPLICAT ORDERS,  $SYSTEM.GGSDAT.ET, EXTSEQNO 26, EXTRBA 1203780
```

Assigning CPUs

When you add a Replicat group you can specify CPUs and a process priority. The options are:

`CPU` *`primary_cpu`*
The processor on which Replicat will run. The default is the CPU on which Manager runs.

`BACKUPCPU` *`cpu`*
An alternative CPU on which to run Replicat if the primary CPU becomes unavailable.

`PRI` *`priority`*
The NonStop priority of the process. Refer to the HP NonStop documentation for more information.

Specifying an alternative Replicat process

The default process name is `$GGRnn`, where `nn` represents the sequence of the process. Oracle GoldenGate recommends that you use the default, however, if you must specify an alternative process, you can do so with the `PROCESS` *`process_name`* option.

Example

```
ADD REPLICAT FINANCE,  $SYSTEM.GGSDAT.ET, PROCESS $GGR04
```

Specifying an alternative parameter or report file

Oracle GoldenGate recommends that you use the default names, however, if you must specify an alternative process, use the options described here. Alternatively, you can change the default names globally from the `GLOBALS` parameter file using `ADD DEFINE`; see "GLOBALS Parameters Summary" for more detail on this parameter. Also see Changing Default Component Names.

- The default parameter file name is *GGS_volume*.GGSPARM.*group_name*, where *group_name* represents a Replicat group, such as FINANCE.
- The default report file name is *GGS_volume*.GGSRPT.*report_name*, where *report_name* represents the report file name, such as FINANCE. Oracle GoldenGate creates an entry-sequenced file to hold each Replicat group's run results. By default, the report name is the same as the Replicat group.

To change the default names:

PARAMS *param_file_name*
Supplies an alternative parameter file name. Enter the fully qualified path name for the parameter file.

REPORT *report_name*
Supplies the new report file name. Enter the fully qualified path name for the parameter file.

Example

This example changes the default parameter file and report names.

```
ADD REPLICAT FINANCE,
 $SYSTEM.GGSDAT.ET,
PARAMS $PARAMS.GGSPARM.FINANCE,
REPORT $REPORTS.GGSRPT.FINANCE
```

Configuring initial data synchronization or other tasks

For initial synchronization or other task processing, you can configure Replicat to run as a task by specifying the SPECIALRUN parameter.

Example

```
ADD REPLICAT group_name, SPECIALRUN
```

Enter a Replicat group description

Use the DESC "*text*" option to describe a Replicat group.

Example

```
ADD REPLICAT T24SEA,  $SYSTEM.GGSDAT.ET,
DESC "T24 data pump for ATM transactions to IBM in Seattle"
```

Executing user exits

You can create and run your own routines by compiling them into an object file and binding this to the Replicat program by using the TACL macro named BINDEXIT. For more information, see User Exits.

When you are ready to call the user exit, launch the Replicat object that has the bound routines with the PROGRAM *program_name* option. Manager uses that program when starting the process.

Example

```
ADD REPLICAT GROUP1, BEGIN NOW, CPU 1, PRI 150, PROGRAM $DATA.GGS.FINEXIT1
```

ALTER REPLICAT

Use `ALTER REPLICAT` to change attributes established in `ADD REPLICAT`.

Syntax

```
ALTER REPLICAT group_name [, options...]
```

group_name
An existing Replicat group name.

options
You can specify any `ADD REPLICAT` option here. If no options are specified, the checkpoint is reset to the beginning of the trail. If `BEGIN` is not specified, the first file in the sequence must exist.

The following example alters the checkpoints for a group of Replicat processes. Use this to skip over data that had not been processed before an unplanned outage.

Example

```
ALTER REPLICAT REP1AP BEGIN NOW
```

CLEANUP REPLICAT

`CLEANUP REPLICAT` deletes old run history records for a group, but keeps the last run record intact, enabling processing to resume from the correct position.

For example: `CLEANUP REPLICAT FINANCE` deletes run history records for the finance group, and keeps the last run record. You can also specify a quantity of records to save, as in: `CLEANUP REPLICAT * SAVE 5`, saving the last 5 run records.

Syntax

```
CLEANUP REPLICAT group_name [SAVE count]
```

group_name
The group name. You can use wildcards to specify a set of group names, such as, `*` or `*FIN*`.

SAVE *count*
Save the last *count* runs.

DELETE REPLICAT

`DELETE REPLICAT` deletes a Replicat group. `DELETE` can have the side effect of freeing up trails for purging by Manager, since associated trail checkpoints are deleted.

Syntax

```
DELETE REPLICAT group_name [!]
```

group_name
The group name. You can use wildcards to specify a set of group names, such as, `*` or `*FIN*`.

!
(Exclamation point) Deletes each group without prompting the user to confirm the operation.

INFO REPLICAT

`INFO REPLICAT` retrieves processing history for a Replicat group. You can specify reporting options to obtain:

- Status of the process
- Process run history
- A lag report
- Detailed historical checkpoints
- Only processes that are running, or stopped

Syntax

```
INFO REPLICAT group_name
[, BRIEF | DETAIL]
[, LAG number {SECONDS | MINUTES | HOURS}]
[, SHOWCH]
[, UP | DOWN]
[, TASKS | ALLPROCESSES]
[, PROGRAM]
```

group_name
The group name. You can use wildcards to specify a set of group names, such as, `*` or `*FIN*`.

BRIEF
Reports the status of the Replicat process (`STARTING, RUNNING, STOPPED` or `ABENDED`) and an approximation of the time and byte lag between the associated trail and Replicat processing.

DETAIL
Reports Replicat process run history, which includes starting and stopping points within the trail expressed as a time, and the process parameters established by the `ADD REPLICAT` command.

LAG *number* SECONDS | MINUTES | HOURS
Restricts the display to groups that are a specified number of seconds, minutes or hours behind. This helps spot critical conditions. The lag returned by this command is approximate. For precise information, use `LAG REPLICAT`. Lag measures both bytes behind and time behind.

SHOWCH
Shows detailed historical checkpoints.

UP | DOWN
Shows processes that are running (`UP`) or not (`DOWN`).

TASKS | ALLPROCESSES
Shows either tasks or all processes that are running. Specify either `TASKS` or `ALLPROCESSES`.

PROGRAM
Displays the name and location of the object that is running.

KILL REPLICAT

`KILL REPLICAT` forces a Replicat process to stop immediately. Try `STOP REPLICAT` first because it also performs cleanup. `STOP` and `KILL` are preferred to stopping from TACL. Manager automatically restarts processes that are stopped from TACL.

Syntax

```
KILL REPLICAT group_name
```

group_name
The group name. You can use wildcards to specify a set of group names, such as, `*` or `*FIN*`.

LAG REPLICAT

Instead of reading the current checkpoint position, `LAG REPLICAT` queries Replicat to determine the relative position of the process in the local trail. This command provides a better estimate of Replicat's lag behind the process than `INFO REPLICAT`.

You can retrieve lag information from remote processes by specifying the Replicat process name instead of group name.

Syntax

```
LAG REPLICAT {group_name | process_name}
```

group_name
The group name, as in: `LAG REPLICAT FINANCE`

process_name
The Replicat process name, as in: `LAG REPLICAT $DATA.GGS.$GGR00`

SEND REPLICAT

`SEND REPLICAT` communicates with a running Replicat process.

Using `SEND REPLICAT` options you can perform a variety of operations that are summarized in "SEND REPLICAT Options Summary".

Syntax

```
SEND REPLICAT group_name {
STATUS |
REPORT [time_option [RESET | FILE name | TABLE name]]|
ROLLREPORT |
LAGSTATS option |
LAGSNAPSHOT |
LAGREPORTON |
LAGREPORTOFF |
LAGOFF |
FORCESTOP |
STOP |
HANDLECOLLISIONS file_name |
```

```
NOHANDLECOLLISIONS file_name |
REPORT HANDLECOLLISIONS |
CLOSEFILES |
GETGROUPTRANSOPS |
SETGROUPTRANSOPS number |
GETMAXTRANSOPS |
SETMAXTRANSOPS number |
GETNETWORKCHECKPOINTS |
    ROLLDISCARD |
ROLLSQLDISCARD
}
```

group_name
A running Replicat group. If the group is not running, an error is returned.

SEND REPLICAT Options Summary

FORCESTOP | STOP
See "Stopping the process".

CLOSEFILES |
Causes Replicat to close any open Enscribe and SQL/MP tables.

REPORT HANDLECOLLISIONS GETGROUPTRANSOPS | SETGROUPTRANSOPS *number* | GETMAXTRANSOPS | SETMAXTRANSOPS *number* | GETNETWORKCHECKPOINTS | HANDLECOLLISIONS *file_name* | NOHANDLECOLLISIONS *file_name*
See "Setting and viewing parameters".

LAGSTATS *option* | LAGSNAPSHOT | LAGREPORTON | LAGREPORTOFF | LAGOFF
See "Obtaining lag reports".

ROLLREPORT |
See "Opening and Closing Discard and Report files".

STATUS | REPORT
See "Obtaining process reports".

ROLLDISCARD | ROLLSQLDISCARD
See "Opening and Closing Discard and Report files"

Obtaining process reports

You can specify reports for:

- Processing status by specifying the `STATUS` option. `STATUS` returns a detailed status of process state, including current position and activity.
- Process statistics by specifying the `REPORT` option. `REPORT` generates an interim Replicat statistical report to the report file, including the number of inserts, updates, and deletes. `Refer to` "SEND REPORT" for detail on `SEND REPORT` options.

Opening and Closing Discard and Report files

To close the current report file and open a new one, specify the `ROLLREPORT` option. `ROLLREPORT` renames the current file to *`report_file1`*, then opens a new report file with the original name. For example, if the original name was `$DATA.GGSRPT.REPCUST`, the `ROLLREPORT` option would rename files: `$DATA.GGSRPT.REPCUST0`, `$DATA.GGSRPT.REPCUST1`,

up to `$DATA.GGSRPT.REPCUST9`. The original report name is recycled and the new report would be named: `$DATA.GGSRPT.REPCUST`.

To close the current discard and open a new one, specify the `ROLLDISCARD` option. To close the current SQL formatted discard file and open a new one, specify the `ROLLSQLDISCARD` option. Like the `ROLLREPORT`, these options rename the current discard file by adding 0 and increment the sequence number of the remaining discard files. If a *`discard_file9`* exists, it will be deleted to make room for the replacement.

> **Note:**
>
> Discard files that have been created by default cannot be rolled over.

Obtaining lag reports

You can obtain a variety of lag reports with the following options:

`LAGSTATS option`
Collects and optionally reports lag statistics. The options are the same as those in the `LAGSTATS` parameter for Replicat. This `LAGSTATS` specification replaces any previous `LAGSTATS` entry.

`LAGSNAPSHOT`
Outputs a report regarding current statistics to the screen and to the report file. To generate this report, set up `LAGSNAPSHOT`, either through the parameter file or dynamically using `SEND`.

`LAGREPORTON`
Generates a report for each lag interval.

`LAGREPORTOFF`
Turns off automatic reporting, but continues to collect data.

`LAGOFF`
Turns off lag statistics.

Setting and viewing parameters

You can set and view the settings for certain parameters.

`GETGROUPTRANSOPS`
Outputs the current number of operations that are grouped together for processing.

`SETGROUPTRANSOPS number`
Sets the number of operations that are to be grouped to the *`number`* value.

`GETMAXTRANSOPS`
Outputs the maximum number of operations that are currently allowed for a transaction.

`SETMAXTRANSOPS number`
Sets the maximum number of operations that can be in a transaction to the *`number`* value.

GETNEWWORKCHECKPOINTS
Outputs the network checkpoint file locations, date of last update, and status information.

HANDLECOLLISIONS *file_name*
Directs Replicat to apply HANDLECOLLISIONS logic. This can also be specified as a startup parameter in the Replicat parameter file. The *file_name* option can be used with or without wildcards to include one or more files. If no *file_name* is specified, HANDLECOLLISIONS will be turned on for all.

NOHANDLECOLLISIONS *file_name*
Directs Replicat to stop applying HANDLECOLLISIONS logic. The *file_name* option can be used with or without wildcards to specify one or more files. If no *file_name* is specified, HANDLECOLLISIONS will be turned off for all.

REPORT HANDLECOLLISIONS
Outputs the status (ON or OFF) of the HANDLECOLLISIONS flag for each file or table.

Examples

Example 1
This example requests the number of operations being grouped for all Replicats.

```
SEND REP *, GETGROUPTRANSOPS
```

The Replicats return:

```
GGRLOG     GROUPTRANSOPS is 50
REPSQL     GROUPTRANSOPS is 100
```

Example 2
This example sets the maximum number of operations that can be in a transaction to 1000.

```
SEND REP REQSQL, SETMAXTRANSOPS 1000
```

Replicat returns:

```
MAXTRANSOPS was set to 1000
```

Example 3
The following command requests information on network checkpoint files.

```
SEND REPLICAT REP01 GETNETWORKCHECKPOINTS
```

This returns a display similar to:

```
Network Checkpoints       Entries 3, Table Size 16
Filename                                 Updated               Fnum  Err  State
--------------------------------- ------------------- ---- ---- -----
\NY.$DATA01.GGS.REPCTXT              2010/01/08 10:43:28      2    0
\LA.$DATA03.GGS.REPCTXT              2010/01/08 10:43:28      3    0
\SEA.$DATA01.GGS.REPCTXT             2010/01/08 10:43:28      4    0
```

Example 4
Sending the first of the following commands turns HANDLECOLLISIONS ON for TCUSTMER. The second requests a report on the settings for HANDLECOLLISIONS.

```
SEND REQSQL, HANDLECOLLISIONS \NY.$DATA4.GGSTAR.TCUSTMER
SEND REQSQL, REPORT HANDLECOLLISIONS
```

The report will be similar to:

```
Reading \NY.$DATA4.GGSDAT.ET000000, Current RBA       2280
Report at 2010-11-10 09:02:39 (Current settings)
Table/File                                            HANDLECOLLISIONS
MAP \LA.$DATA4.GGSSOU.TCUSTMER
       to \NY.$DATA4.GGSTAR.TCUSTMER                      On
MAP \LA.$DATA4.GGSSOU.TCUSTORD
       to \NY.$DATA4.GGSTAR.TCUSTORD                      Off
```

Stopping the process

You can stop the current process using either the `FORCESTOP` or `STOP` option.

`FORCESTOP`
Instructs Replicat to rollback the pending transaction and stop the process immediately.

`STOP`
Terminates Replicat gracefully. This command is preferable to stopping Replicat from TACL or other command prompt, which results in an `ABEND` status.

START REPLICAT

Use `START REPLICAT` to begin a Replicat process. Manager receives the request to start and monitor the process.

Syntax

```
START REPLICAT group_name
```

`group_name`
The group name. You can use wildcards to specify a set of group names, such as, `*` or `*FIN*`.

STATUS REPLICAT

Use `STATUS REPLICAT` to determine whether Replicat processes are running.

Syntax

```
STATUS REPLICAT group_name [, DETAIL | TASKS | ALLPROCESSES]
```

`group_name`
The group name. You can use wildcards to specify a set of group names, such as, `*` or `*FIN*`.

`DETAIL`
If `DETAIL` is specified, (`STATUS REPLICAT *, DETAIL`) the audit trails required by the Replicat group are also displayed. Output consists of the locations of required audit trails, whether they are on disk or tape, and whether they still exist.
`DETAIL` is useful for determining:

- Whether audit must be restored from tape before the group is run
- Which Replicat groups are causing Manager to tie up TMF resources

TASKS | ALLPROCESSES
Provides status on tasks or all processes that are running. Specify either TASKS or ALLPROCESSES.

STOP REPLICAT

STOP REPLICAT stops a Replicat process gracefully. Using this command lets you make configuration changes without affecting the operation of future runs, and ensures that Manager will not restart the process.

Syntax

```
STOP REPLICAT group_name [, WAIT [seconds] | ATEND |!]
```

group_name
The group name. You can use wildcards to specify a set of group names, such as, * or *FIN*.

WAIT *seconds*
GGSCI waits for the process to terminate before issuing the next prompt. When a value is specified for *seconds*, GGSCI waits up to that many seconds before returning control to the user. If you do not specify WAIT, GGSCI issues the next prompt immediately.

ATEND
Instructs Replicat to terminate when it reaches end-of-file in the last sequence of trails. Replicat also terminates if the trail is no longer available (due to network outage, or other condition). ATEND guarantees that all outstanding records have been processed.
The current transaction is rolled back if the trail contains only part of the last transaction. If the last transaction was part of a grouped transaction (GROUPTRANSOPS parameter) and the source trail is available, the individual transactions are replayed up to the point where Replicat is quitting.

!
Stops Replicat even in the middle of a transaction. Use this option to terminate long running transactions. As with ATEND, a grouped transaction is rolled back but the individual transactions are replayed if the trail is available.

ER Commands

Oracle GoldenGate lets you manage Extract and Replicat as a unit with a single command. For example, to start the modules separately for group FINANCE, you would normally enter commands similar to:

```
GGSCI> START EXTRACT EXTFIN
GGSCI> START REPLICAT REPFIN
```

Using combined management, you can start both modules with a single command, as in:

```
GGSCI> START ER *FIN
```

Syntax

```
command ER group_name [, option]
```

command
Any one of the following:

INFO
Returns the processing status of both modules, including lag information

KILL
Forces the processes to stop immediately. Oracle GoldenGate recommends first attempting to stop processes using the `STOP` command. Either `KILL` or `STOP` is preferred over stopping processes from TACL.

SEND
Sends a performance *option* to the programs, such as `REPORT`.

```
SEND ER *FIN, REPORT
```

The `REPORT` option generates an interim statistical report to the report files.

START
Begins the programs. Manager receives the `START` command and starts and monitors the programs.

STATUS
Determines whether the processes are running.

STOP
Causes a graceful stop, ensuring configuration changes can be made without impacting future runs.

For more information about these commands, see the command's description in "Extract commands" or "Replicat commands".

ER
Required. Informs Oracle GoldenGate that the command applies to both of the programs.

group_name
The Extract or Replicat group name. You can use wildcard specifications. The following commands act upon any group containing the characters `FIN`.

```
START ER *FIN*
START ER FIN*
START ER *FIN
```

option
Can be any option associated with the command, such as the `INFO` command `DOWN` option, which shows only processes that are not running, as in:

```
INFO ER *FIN, DOWN
```

For details about the options, see the command's description in "Extract commands" or "Replicat commands".

SEND ER* STATS

Use the following commands to get process status from Extract and Replicat, as in:

Syntax

```
SEND [EXTRACT | REPLICAT] group_name,
STATS [time_option [RESET | FILE name | TABLE name]]
```

group_name
The group name you defined with the appropriate GGSCI `ADD` command

`STATS` ***`time_option`*** **`[RESET | FILE`** ***`name`*** **`| TABLE`** ***`name`***
Returns Extract and Replicat processing statistics based on the selected options.

`RESET`
This resets the *`time_option`* counters to zero. For example, the command `SEND RECENT RESET` will report the `RECENT` counters and then reset them to zero.

`FILE` ***`name`*** **`| TABLE`** ***`name`***
This limits the display to statistics for the file or table specified in *`name`*.

`time_option`
Each of these returns processing statistics for the specified process. `TRANSACTION` is valid only for Replicat.
`TOTALS`: since the Extract or Replicat was started
`DAILY`: since the beginning of the current day
`HOURLY`: since the beginning of the current hour
`RECENT`: since the last time the `RECENT` counter was reset
`TRANSACTION`: since the beginning of the current transaction

Examples

NOT_SUPPORTED
The following sample Extract report uses the default `STATS` option.

```
Report at 2010-02-27 12:43:07 (activity since 2010-02-27 11:43:44)
Elapsed time 0-00:59:22.916658
Total # records written to \NODEA.$TEST04.GGSDAT.ET               19
   \NODEA.$TEST04.GGSSOU.TCUSTMER           #  inserts:            3
                                            #  updates:            0
                                            #  deletes:            0
   \NODEA.$TEST04.GGSSOU.TCUSTORD           #  inserts:            3
                                            #  updates:           13
                                            #  deletes:            0
```

NOT_SUPPORTED
The next sample Extract report uses the following options.

```
TOTALS statistics since 2018-07-12 10:37:53
                #  inserts:         1
                # purgedatas:         1
DAILY  statistics since 2018-07-12 10:37:53
                #  inserts:         1
                 # purgedatas:         1
HOURLY statistics since 2018-07-12 10:59:59
```

```
               #  inserts:         1
               # purgedatas:         1
RECENT statistics since 2018-07-12 10:37:53
               #  inserts:         1
               # purgedatas:         1
```

NOT_SUPPORTED
The following Replicat report uses the `TRANSACTION` option.

```
Report at 2010-02-28 06:51:15 counters for TRANS since 2010-02-14 13:46:54
Elapsed time 13-17:04:21.290242
From \A.$TEST04.GGSSOU.ECUSTMER to \A.$TEST04.GGSTAR.HCUSTMER:
           #  inserts:         3
           #  updates:         1
           #  deletes:         0
           # discards:         0
From \A.$TEST04.GGSSOU.ECUSTORD to \A.$TEST04.GGSTAR.HCUSTORD:
           #  inserts:         3
           #  updates:         3
           #  deletes:         2
           # discards:         0
```

Logger commands

Use Logger commands to configure Logger for extracting data changes from non-TMF applications.

Logger processes default to a prefix of `$GGL`. To change these default settings, see Change Synchronization for Non-TMF Applications.

ADD LOGGER

Use `ADD LOGGER` to configure `GGSLIB` and Logger. By default, `ADD LOGGER` reads a parameter file called *`GGS_volume`*`.GGSPARM.LOGPARM`. Before invoking `ADD LOGGER`, edit `LOGPARM` to enter the appropriate parameters. See "Logger Parameters Summary" for information about the parameters you can enter into this file.

To bind `GGSLIB` to the application, see "Programs commands".

Syntax

```
ADD LOGGER [, PARAMS param_file_name]
```

`PARAMS param_file_name`
Use the `PARAMS` option to indicate a different parameter file name.

`ADD LOGGER` performs the following:

- Validates the configuration parameters for Logger.
- Creates a *segment file* containing parameters used by `GGSLIB` intercept library routines. These parameters tell the intercepts where to send logged information (that is, which Loggers should receive it).

  The default segment file is `$SYSTEM.GGS.AUDCFG`. It is strongly recommended that you use the default location. If you must use a different location (such as for running multiple occurrences of the Oracle GoldenGate environment), see Changing Default Component Names.

If the segment file exists at the time ADD LOGGER is issued, GGSCI renames the existing segment and issues a message informing you the existing AUDCFG is renamed.

- Creates log trail files and pre-allocates space for the log trails. GGSCI has a limit of 200 log trails per Logger process. The maximum number of Logger processes per instance is 50.
- Updates the GGS database to recognize the Loggers and configuration parameters.

ALTER LOGGER

Use ALTER LOGGER to change parameters for an existing Logger process. As with the ADD LOGGER command, ALTER LOGGER reads and validates the parameters in LOGPARM and pre-allocates log trail files.

Syntax

```
ALTER LOGGER [, PARAMS param_file_name]
```

PARAMS *param_file_name*
Use the PARAMS option to indicate a different parameter file name.

Configuration segment files are aged every time ALTER LOGGER is invoked (AUDCFG00, AUDCFG01, and so on). If the segment file exists at the time ALTER LOGGER is issued, GGSCI renames the existing segment and issues a message informing you the existing AUDCFG is renamed. ALTER LOGGER renames up to 99 files.

Logger parameter changes take effect immediately. Altering Logger parameters while Logger or Replicat processes are running requires careful consideration. In particular, consider the situation in which a file set is switched from one Logger to another.

If you execute commands for ALTER LOGGER that change what files are logged to which Logger process, expect to see a message similar to the following:

```
14-11-12;12:23:28:22.964 \NY,3,28      ORACLE.1.H06           512
            FILE \NY.$DATA04,TEST1.ABC switched to logger \NY.$KK00
```

DELETE LOGGER

Use DELETE LOGGER to delete the internal Oracle GoldenGate Logger configuration files created when you entered ADD LOGGER.

Optionally, you can also delete the Logger trails with the ! (exclamation point) option. Before executing this option, ensure that all of the data has been processed out of the trail.

Syntax

```
DELETE LOGGER [!]
```

!
The exclamation point (!) deletes files from the associated log trail. The log trails associated with the process must be manually deleted if you omit the exclamation point or they will still exist.

INFO LOGGER

Use `INFO LOGGER` to retrieve the following information about Logger:

- The location of the shared configuration segment used by `GGSLIB`
- The date Logger was added and the location of the parameter file used to create it
- The settings for timeout and debug, and whether logging is currently on
- Which sequence number Logger currently has open for each Logger process
- The location of the trail used by each Logger process
- Whether each Logger process in the group is running
- `CPU, BACKUPCPU` and `PRIORITY` values for each process
- Configured flushing and tracing parameters
- The block size used to write the trail
- Whether the trail is version 7 format (New) or the format from before version 7 (Old)
- `FILE` entries for each Logger process, along with the settings for the following `FILE` configuration options for each file
  - Whether the image is compressed for updates
  - Whether TMF audited file changes are logged
  - Whether unstructured file changes are logged
  - Whether Logger includes bulk I/O updates
  - The delay interval to detect and record a new name
  - The Log Mode of the file (suspended or active)

Syntax

```
INFO LOGGER
[, AUDCFG segment_file]
[, SHOWLOGGED file_name
[, PROGRAM program_set | PROCESS process_set | USER user_set]]
[, BRIEF]
```

AUDCFG *segment_file*
Determines the parameters for any configuration, including the current one. Configuration segment files are aged every time `ALTER LOGGER` is invoked (`AUDCFG00`, `AUDCFG01`, and so on).

BRIEF
Limits the `INFO LOGGER` display to information about the `SHOWLOGGED` file name.

SHOWLOGGED *file_name*
Lets you determine which log process, if any, is capturing a particular file. Specify the file name or wildcard file set. Whether the file is included, excluded or omitted for each Logger is displayed.
If you specify `SHOWLOGGED`, you can also detect whether a file is included or excluded according to one of the following:

PROCESS *process_set*
Directs Logger to extract data only when the opener is the process or set of processes specified. (The process set can be a single process or a wildcard, for example `$APP*`.)

PROGRAM *program_set*
Directs Logger to extract data only when the opener is the program or set of programs specified. (The program set can be a single program or a wildcard, for example `$DATA1.PROGS.*`.)

USER *user_set*
Directs Logger to extract data only when the creator access ID of the opener is the user specified. (The user set can be a single user, for example `FINANCE.JOE`, or a wildcard, for example `SUPER.*`.)

Example

Sample results from an `INFO LOGGER` command:

```
Information for Logger Group $GGL

Intercept segment file \NY.$SYSTEM.GGS.AUDCFG
Created  2010-10-21 11:24
Built from \NY.$DATA01.GGSPARM.LOGPARM
Logger timeout: 60.00 seconds
Debug on stack check: Off
Current mode: Logging is ON

  Process:        $GGL49
  Log Trail:      \NY.$DATA01.LOGGER.ET000013
  Status:         DOWN
  CPUs:           2,3
  Priority:       170
  Logopens:       16
  Flush recs:     16
  Flush secs:     0.85
  Trace IOs:      Off
  Trace Stats:    Off
  Heartbeat:      Off
  AdjustPriority: On
  BlockSize:      57344
  TrailFormat:    New
  SourceAppInfo:  Included
  Logger Timeout: 60.00 seconds
                                  Comp   Unstr TMF    Bulk  Rename Log
  Files                           Updts  Files Files  IO    Delay  Mode
  \NY.$D*.*.*                      No     Yes    No     Yes   No     Normal
```

SEND LOGGER

Use `SEND LOGGER` to communicate with one or more running Logger processes.

Syntax

```
SEND LOGGER

[, PROCESS process_name]
[, ADJUSTPRIORITY | NOADJUSTPRIORITY]
[, ROLLOVER]
```

```
[, REFRESH]
[, PROCESSINFO [, DETAIL]]
[, LOGFILECLOSEDELAY seconds]
[, LOGINFO]
[, FLUSHSTATS
   [, FILTERPROCESS process_name |
      FILTERPROGRAM program_name |
      FILTERLIBRARY library_name] |
   [, RESET]]
[, GETSTATS
   [, FILTERPROCESS process_name |
      FILTERPROGRAM program_name |
      FILTERLIBRARY library_name] |
   [, RESET]]
[, GETLOGFILECLOSEDELAY]
[, HOTSWAP object_name]
```

PROCESS *process_name*
Sends to the named process. Otherwise, the command is sent to all Logger processes in the default Logger group (`$GGL*`).

ADJUSTPRIORITY | NOADJUSTPRIORITY
Determines how Logger adjusts its priority in relation to the sender priority.
Logger checks at 1 minute intervals to determine if there was a high priority sender during the previous interval. If not, by default, Logger sets its priority back down to the original value.
To retain the value set by the sender, specify `ADJUSTPRIORITY`, as in:

```
GGSCI SEND LOGGER, ADJUSTPRIORITY
```

If `NOADJUSTPRIORITY` is in effect Logger does not increase its priority to match that of a higher priority sender.

ROLLOVER
Instructs Logger to move to the next log file in the log trail sequence.

REFRESH
Instructs Logger to read its `LOGCONF` record to pick up configuration option changes.

PROCESSINFO
Instructs Logger to return input and output statistics, lag, and other information to the screen. Optionally, include `DETAIL` to report process details.

LOGFILECLOSEDELAY
Sets the time to delay closing the old file when rolling to a new one. Logger starts the timer, writes to the new file and closes the old file when the timer is reached or when it rolls to another new file. The default value is 120 seconds.

LOGINFO
Displays information on Logger's current log file.

FLUSHSTATS
Instructs Logger to output current process statistics to the log trail. Optionally, include `RESET` to reset all statistics counters to zero.
You can specify one of the following filters per `SEND LOGGER FLUSHSTATS` command.

- `FILTERPROCESS` *`process_name`*

- FILTERPROGRAM *program_name*
- FILTERLIBRARY *library_name*

These allow you to restrict flushed information to the specified process, program, or library. The name is the specified process, program, library, or a wildcard. Use only one of these options.

GETSTATS
Outputs current process statistics to the screen.
If you specify GETSTATS, you must specify either TRACESTATS or TRACEPROCESSIOS in the Logger parameter file.
Optionally, include RESET to reset all statistics counters to zero.
You can specify one of the following filters per SEND LOGGER GETSTATS command.

- FILTERPROCESS *process_name*
- FILTERPROGRAM *program_name*
- FILTERLIBRARY *library_name*

These allow you to restrict flushed information to the specified process, program, or library. The name is the specified process, program, library, or a wildcard. Use only one of these options.

GETLOGFILECLOSEDELAY
Outputs the current value for LOGFILECLOSEDELAY.

HOTSWAP *object_name*
Instructs Logger to use the specified object file. Allows a running Logger to be changed to a different Logger object.
Caution: The HOTSWAP command must be performed with the manual steps used to upgrade BASELIB as part of the upgrade process.

START LOGGER

Use START LOGGER to start a group of Logger processes. By default, this group is $GGL. If some log processes are running, START LOGGER begins the ones that are down.

Syntax

```
START LOGGER [, NAME process_name]
```

NAME *process_name*
The name of the Logger process, such as $GGL. To start a particular process within the group name specify the full name of the process, such as $GGL01.

STATUS LOGGER

Use STATUS LOGGER to obtain process status for the specified Logger process.

Syntax

```
STATUS LOGGER [,NAME process_name]
```

NAME *process_name*
Provides the status of a particular process within the group, such as $GGL01.

STOP LOGGER

Use this command to stop a group of Logger processes. By default, this group is `$GGL`.

If some log processes are down, `STOP LOGGER` brings down the remainder. Use this command cautiously, since no data is logged while log processes are down. When issuing this command, you will be prompted to specify whether you want to continue.

Use `STOP LOGGER` instead of stopping log processes individually from TACL. By default, Manager restarts log processes stopped from TACL.

Syntax

```
STOP LOGGER [, NAME process_name] [!]
```

NAME *process_name*
To stop a particular process within the group, specify the full name of the process (for example, `$GGL01`).

!
When you issue `STOP LOGGER`, with or without options, you are prompted to confirm the operation. To override the prompt, include `!` (exclamation point) in the command argument.

Trail commands

Trail commands allow you to create and associate a sequence of local or remote trails with a particular Extract group. This is particularly useful in online processing to purge or transfer old Oracle GoldenGate trails without bringing down the associated Extract process.

Use `RMTTRAIL` commands to create and manage trails on remote systems, and use commands to create and manage local Oracle GoldenGate trails.

ADD

Use `ADD` to create a local trail, associate it with an Extract group, and assign trail attributes. If the trail already exists, GGSCI rejects the `ADD` command.

Syntax

```
ADD  trail_name, EXTRACT group_name
[, OWNER group_number, user_number] [, SECURE "rwep"]
[, EXTENTS (primary, secondary, maximum) | MEGABYTES number]
[, MAXFILES num_files]
[, SEQNO number], [, CREATE | NOCREATE]]
```

trail_name
The fully qualified trail name: $*vol*.*subvol*.*trail_prefix*. The *trail_prefix* must be two characters long. Each file in the trail is automatically identified by the prefix and a six-digit serial number. The parameter file for *group_name* must have a matching *trail_name* parameter.

EXTRACT *group_name*
Specifies the Extract group to which the is bound. Only one group can write to an associated trail.

EXTENTS (*primary*, *secondary*, *maximum*) | MEGABYTES *number*
See "Specifying file size".

MAXFILES *num_files*
See "Specifying a maximum number of files".

OWNER *group_number*, *user_number* SECURE "*rwep*"
See "Specifying security".

SEQNO *number*
See "Specifying sequence number".
See Ongoing Trail Managementfor more information on managing trails.

CREATE | NOCREATE
To specify whether the file is created, see "Using a Local Oracle GoldenGate Trail"".

Specifying file size

Control file size with one of the following options:

- Use the `EXTENTS` *`primary`*, *`secondary`*, *`maximum`* option to specify extent sizes for individual trails. Default extent sizes are 64, 128 and 512.
- Use the `MEGABYTES` *`number`* option to specify the maximum number of megabytes per file in the trail. The default is 134 megabytes, and the maximum is 2000. To allow the Extract `ROLLOVER` parameter to determine when new files are created, set *`number`* to a large number, such as 1000 megabytes.

Examples

Example 1
This example adds a trail with a maximum size of 300 megabytes.

```
ADD  $DATA.GGSDAT.EX, EXTRACT FINANCE, MEGABYTES 300
```

Example 2
This example adds a trail with the extents set to `10` for primary and secondary and `16` for the maximum.

```
ADD  $DATA1.EXTDAT.EX, EXTRACT FINANCE, EXTENTS (10,10,16)
```

Specifying a maximum number of files

Use the `MAXFILES` *`num_files`* option to specify the maximum number of files that can exist in a trail. The default for `MAXFILES` is 100.

Example

This example adds a trail with a maximum of 20 files.

```
ADD  $DATA1.EXTDAT.EX, EXTRACT FINANCE, MAXFILES 20
```

Specifying security

Specify security measures to restrict access to Oracle GoldenGate trails. If you do not specify security, the defaults are assumed.

- Use `OWNER` *`group_number`*, *`user_number`* to specify the NonStop group ID and user ID of the person who started the GGSCI process.
- Use `SECURE "rwep"` to specify the default Guardian security attributes (read, write, execute, purge) of the person who started the GGSCI process.

Example

This example specifies a trail that is owned by user `100, 23` and can be read by anyone in the network in group `100`.

```
ADD  $DATA1.EXTDAT.EX, EXTRACT FINANCE, OWNER 100,23, SECURE "CUUU"
```

Specifying sequence number

Specify the trail sequence number for the first file in the trail. Do not include any zero padding.

Example

The following example specifies that the first file in the trail will be `ex000003`.

```
ADD  $DATA1.EXTDAT.EX, EXTRACT FINANCE, SEQNO 3
```

Use `SEQNO` during troubleshooting when Replicat must be repositioned to a certain trail sequence number. This eliminates the need to alter Replicat to read the required sequence number.

ADD RMTTRAIL

`ADD RMTTRAIL` creates a remote Oracle GoldenGate trail on a remote system, assigns a maximum size to each file, and associates the file with a particular group. Rolling over from one sequence to the next can be controlled using the maximum size allowed, or by using the `EXTRACT ROLLOVER` startup parameter or `ETROLLOVER` option of the GGSCI `ALTER EXTRACT` command.

In the parameter file, specify a `RMTHOST` entry before any `RMTTRAIL` entries to identify the remote system and TCP/IP port.

Syntax

```
ADD RMTTRAIL trail_name, EXTRACT group_name
[, MEGABYTES number]
[, SEQNO number]
```

`trail_name`
The fully qualified trail name: *`$vol.subvol.trail_prefix`*. The *`trail_prefix`* must be two characters long. Each file in the trail is automatically identified by the prefix and a six-digit serial number. The *`trail_name`* must have a matching entry in the Replicat parameter file.
The name you specify here must be the same name you specify for the `RMTTRAIL` parameter in the Extract parameter file.

Remote trails are used over TCP/IP connections only. Do not use when you are transmitting a trail over Expand even when the remote system is also connected with Expand. To specify a trail on a different NonStop node over an Expand connection, use ADD .

EXTRACT *group_name*
The group to which the RMTTRAIL is bound. Only one group can output extracted data to each trail.

MEGABYTES *number*
Optional. The maximum number of megabytes per file in the trail. The default is 30 megabytes and the maximum is 2000. To allow the Extract ROLLOVER parameter to determine when new files are created, set *number* to a large number, such as 1000 megabytes.

SEQNO *number*
Optional. Specifies the trail sequence number for the first file in the trail. Do not include any zero padding. Use SEQNO during troubleshooting when Replicat must be repositioned to a certain trail sequence number. It eliminates the need to alter Replicat to read the required sequence number.

Examples

Example 1
The following example illustrates how to add three remote Oracle GoldenGate trails; the first trail residing on UNIX, the second on a Windows platform, and the third on NonStop.

```
ADD RMTTRAIL /usr/extdat/xx, EXTRACT FINANCE, MEGABYTES 30
ADD RMTTRAIL c:\ggsdat\ex, EXTRACT FINANCE, MEGABYTES 30
ADD RMTTRAIL $DATA.GGSDAT.RT, EXTRACT FINANCE, MEGABYTES 30
```

Example 2
The following example specifies that the first file in the trail will be rt000003.

```
ADD RMTTRAIL $DATA1.GGSDAT.RT, EXTRACT FINANCE, SEQNO 3
```

ALTER

ALTER changes attributes for an existing .

Syntax

```
ALTER  trail_name, [, options]
```

trail_name
The fully qualified trail name: *$vol.subvol.trail_prefix*. The *trail_prefix* must be two characters long. Each file in the trail is automatically identified by the prefix and a six-digit serial number.

options
ALTER can be used with the following options:

- EXTENTS (*primary*, *secondary*, *maximum*) | MEGABYTES *number*
- MAXFILES *num_files*

- OWNER *group_number*, *user_number* SECURE "*rwep*"

SEQNO is not a valid option for ALTER .

Example

```
ALTER  $DATA1.EXT1.AA, MAXFILES 50
```

ALTER RMTTRAIL

ALTER RMTTRAIL changes attributes for an existing RMTTRAIL.

Syntax

```
ALTER RMTTRAIL trail_name [, MEGABYTES number]
```

trail_name
The fully qualified name of the remote trail: $*vol.subvol.trail_name*. A six-digit serial number will be appended to each file in the trail.
Remote trails are used over TCP/IP connections only. Do not specify an Expand node name in the *trail_name*, even if the remote system is also connected with Expand. To specify a trail on a different NonStop node over an Expand connection, use ADD .

MEGABYTES *number*
The maximum number of megabytes per file in the trail. The default is 30 megabytes, and the maximum is 2000. To allow the Extract ROLLOVER parameter to determine when new files are created, set *number* to a large number, such as 1000 megabytes.

DELETE

DELETE deletes checkpoints for a specified trail. It does not delete the trail or the files in the trail.

Syntax

```
DELETE  trail_name [!]
```

trail_name
The fully qualified trail name: $*vol.subvol.trail_prefix*, or a wildcard specification, as in:

```
DELETE  $DATA1.EXTDAT.AA
DELETE  *
```

!
(exclamation point) Deletes trail files associated with each group.

DELETE RMTTRAIL

DELETE RMTTRAIL deletes checkpoints for a particular remote trail. It does not delete the files in the remote trail.

Syntax

```
DELETE RMTTRAIL trail_name
```

trail_name
The fully qualified name of the remote Oracle GoldenGate trail as in: `DELETE RMTTRAIL /usr/dat/aa.` A six-digit serial number will be appended to each file in the trail.
Remote trails are used over TCP/IP connections only. Do not specify an Expand node name in the *trail_name*, even if the remote system is also connected with Expand.

INFO

`INFO` retrieves configuration information about the trail.

Syntax

```
INFO  trail_name
```

trail_name
The fully qualified trail name: $*vol*.*subvol*.*trail_prefix*, or a wildcard specification, as in:

```
INFO  $DATA1.EXTDAT.AA
INFO  *
```

INFO RMTTRAIL

`INFO RMTTRAIL` retrieves configuration information for the remote trail.

Syntax

```
INFO RMTTRAIL trail_name
```

trail_name
The name of the Oracle GoldenGate trail. *trail_name* must be a fully qualified file name, as in:

```
INFO RMTTRAIL $DATA6.GGSDAT.BB
```

Remote trails are used over TCP/IP connections only. Do not specify an Expand node name in the *trail_name*, even if the remote system is also connected with Expand. Use `INFO` instead.

Database commands

Use database commands to get information about data definitions and tables.

CAPTURE TABLEDEFS

`CAPTURE TABLEDEFS` returns information for SQL tables and for Enscribe files when you provide a `DICTIONARY` name and `RECORD` definition name.

Syntax

```
CAPTURE TABLEDEFS file_name
[, DICTIONARY volume.subvol]
[, RECORD record_name]
[, OPTIONS command_line_options]
```

file_name
The fully qualified name of the file or table.

DICTIONARY *volume.subvol*
The volume and subvolume of the Enscribe dictionary. Required for Enscribe files.

RECORD *record_name*
The name of the record definition within the Enscribe dictionary. Required for Enscribe files.

OPTIONS *command_line_options*
Valid DEFGEN command-line options. See the chapter on DEFGEN arguments for more information.

> **Note:**
>
> If the OPTIONS argument contains EXPANDDDL parameters, then it overrides the default EXPANDDDL parameters. Currently the default EXPANDDDL parameters are:
>
> ```
> EXPANDDDL
> EXPANDGROUPARRAYS
> NOFIXLONGNAMES
> MAXCOLNAMELEN 130
> ```

Example

```
CAPTURE TABLEDEFS \PROD.$DATA1.ACCTS.KEYSEQ
```

The result of the example command is the following display:

```
Definition for table \PROD.$DATA1.ACCTS.KEYSEQ
Record length: 198
Syskey: 0
Columns: 13

   TS           LARGEINT  PK
   RECNUM       INT       PK
   SYSNAME      CHAR      (8)
   TEXT         CHAR      (64)
   VAL1         LARGEINT
   VAL2         LARGEINT
   COL_COMPUTE  LARGEINT
   I16          SMALLINT
   I32          INT
   I64          LARGEINT
   I32_TOTAL    INT
   JTS          LARGEINT
   JTS_TEXT     CHAR      (64)
```

ENCRYPT PASSWORD

Use ENCRYPT PASSWORD to encrypt a login password for an Oracle GoldenGate database user and, optionally, supply an encryption key for password lookup. To specify the encrypted password in a parameter file, use the LOGON parameter (see "LOGON").

For more information about Oracle GoldenGate security, see Encrypting a Database Password.

Syntax

```
ENCRYPT PASSWORD password [ENCRYPTKEY {DEFAULT | keyname}]
```

password
The login password. The encrypted password is output to the screen. You can copy the encrypted password and paste it into the `LOGON` parameter in a parameter file.

`ENCRYPTKEY {DEFAULT | keyname}`
Optional, specifies one of the following:

`DEFAULT`
Specifies a default encryption key that is randomly generated by Oracle GoldenGate and automatically decrypted on the target system.

keyname
Specifies an encryption key contained in the `ENCKEYS` lookup file. Oracle GoldenGate uses the key name to look up the actual password in the file. To use the *keyname* option, you must create the `ENCKEYS` file on each system (if it does not exist) and create entries in the file for the keys.

Example

```
ENCRYPT PASSWORD ny14072 ENCRYPTKEY superkey2
```

INFO DDLDEFS

Use `INFO DDLDEFS` to retrieve information for Enscribe data dictionary definitions.

Syntax

```
INFO DDLDEFS def_name [, DDLDEFS def_name,...,]
DICT subvolume
[, DEFSONLY | RECSONLY]
```

def_name
The name of a DDL definition or record. You can specify *def_name* multiple times to display multiple definitions or records. Wildcards are accepted.

`DICT` ***subvolume***
The subvolume in which the dictionary is located.

`DEFSONLY`
Specifies that GGSCI should return only definitions, not records.

`RECSONLY`
Specifies that GGSCI should return only records, not definitions.

Example

This example lists each DDL record or definition which begins with `ACC`, or which ends in `REC` from the dictionary located in `$DATA3.MYDICT`.

```
INFO DDLDEFS ACC*, DDLDEFS *REC, DICT $DATA3.MYDICT
```

INFO FILES

Use `INFO FILES` to retrieve information about files or tables on the system; then filter the resulting list according to different criteria.

Syntax

```
INFO FILES file_name, [FILES file_name,...]
[,TMF |
NONTMF |
ENSCRIBE |
SQL |
EXCLUDELASTDIGIT |
CODE file_code |
TANDEMFILES |
UNSTRUCT]
```

file_name
A file name or wildcard specification. Multiple entries of *file_name* are allowed.

TMF
Returns TMF audited files.

NONTMF
Returns files not audited by TMF.

ENSCRIBE
Returns Enscribe files.

SQL
Returns NonStop SQL tables.

EXCLUDELASTDIGIT
Excludes file names which end in a digit. Use `EXCLUDELASTDIGIT` to filter out alternate key files that end in a digit.

CODE *file_code*
Returns files with *file_code* only. Multiple entries of `CODE` are allowed.

TANDEMFILES
Returns NonStop files (file codes between 1 and 1000) only.

UNSTRUCT
Restricts the list to unstructured files.

Audit trail commands

Use `ATCONFIG` commands to protect TMF audit trails until Extract has processed them. Manager uses `ATCONFIG` commands to determine how to preserve audit files that are needed by Extracts.

The `ATCONFIG` command specification can be abbreviated as `AT`.

For details about managing audit resources, see Keeping Necessary Audit Available for Extract.

ADD ATCONFIG

Use ADD ATCONFIG to configure audit management options. With the ADD ATCONFIG options, you can:

- Duplicate to an alternative subvolume
- Duplicate all audit files or a specified number of files
- Purge audit trails from the alternative subvolume

You can override any previously specified option by adding NO, as in NO PURGE.

> **Note:**
>
> Contact Oracle GoldenGate support before using the DUP, DUPFILES, or PURGE options. These options require storage that is not necessary if large enough audit trails can be specified when TMF is configured.

Syntax

```
ADD ATCONFIG at_name
[, ALTLOC alt_subvolume]
[, DUP | NO DUP |
DUPFILES num_files | NO DUPFILES |
PURGE | NO PURGE]
```

at_name
The audit trail designation, that is MAT, AUX*nn*. The audit trail can also be expressed as a wildcard.

ALTLOC *alt_subvolume*
Identifies an alternative subvolume to which audit trails are duplicated or restored from tape. ALTLOC directs Extract to read audit from the alternative subvolume rather than from the production area.
Specify up to seven characters for the volume name.

DUP | NODUP
Duplicates audit files to the volume specified by ALTLOC. DUP has no effect if the file already exists as a TMF disk dump. DUP copies audit files that are still needed by an Extract group.

DUPFILES *num_files* | NO DUPFILES
Duplicates up to *num_files* audit files to the volume specified by ALTLOC. Unlike DUP, DUPFILES limits the number of files that can be copied to the alternative subvolume. When *num_files* is reached on the alternative subvolume, the oldest audit file is purged to make room for the newest file. Enter ALTLOC when this option is used.

PURGE | NOPURGE
Purges audit trails from the alternative subvolume when they are no longer needed. PURGE has no effect when DUPFILES is specified since DUPFILES keeps a constant number of backup files.

Example

The following examples show the MAT trail being added and an AUX trail being altered.

```
ADD AT MAT DUPFILES 6, ALTLOC $DATA1.ALTTMF
ALTER AT AUX01, ALTLOC $DATA1.EXTRACT, PURGE, DUP
```

ALTER ATCONFIG

Use ALTER to change existing audit trail configuration parameters.

Syntax

```
ALTER ATCONFIG at_name
[, ALTLOC alt_subvolume]
[, DUP | NO DUP |
DUPFILES num_files | NO DUPFILES |
PURGE | NO PURGE]
```

See the ADD ATCONFIG command for option descriptions.

DELETE ATCONFIG

Use DELETE ATCONFIG to delete audit management configuration. After this command is carried out, Manager will not manage the associated audit trails.

Syntax

```
DELETE ATCONFIG at_name
```

at_name
The audit trail designation, that is MAT, AUX*nn*. The audit trail can also be expressed as a wildcard.

INFO ATCONFIG

Use INFO ATCONFIG to view processing information about audit trails that are defined by ADD ATCONFIG or ALTER ATCONFIG.

Syntax

```
INFO ATCONFIG at_name
```

at_name
The audit trail designation, that is MAT, AUX*nn*. The audit trail can also be expressed as a wildcard.

STATUS AUDITTRAIL

Use STATUS AUDITTRAIL to determine which audit trail files are still required by any Extract group. This command determines if a file exists, then supplies:

- The location of the file.
- Whether it is the original audit file (ORIG), a duplicate audit file (DUP), or a dump (DUMP).

- Whether the file is on tape or disk. If audit is on tape, this provides the information needed to restore dumps before processing — useful when an operator is not available while Extract processes are running.

Syntax

```
STATUS AUDITTRAIL at_name
```

at_name
The audit trail designation, that is `MAT`, `AUX`*nn*. The audit trail can also be expressed as a wildcard.

Remote checkpoint commands

Use `REMOTECHKPT` when both of the following apply:

- The Manager process is configured to perform local trail maintenance using checkpoints
- The local trail is being processed by programs on remote systems

If the `PURGEOLDEXTRACTS` parameter is set, Manager periodically examines checkpoint files and purges files that satisfy the rules of the `PURGEOLDEXTRACTS` parameter.

ADD REMOTECHKPT

Use `ADD REMOTECHKPT` to tell Manager where to find checkpoints for Extract or Replicat processes running on remote systems. The Replicat checkpoint file is called `REPCTXT` and exists in the same subvolume where Oracle GoldenGate is installed, as in:

```
ADD REMOTECHKPT \NY.$DATA5.GGS.REPCTXT
```

Syntax

```
ADD REMOTECHKPT checkpoint_file
```

checkpoint_file
The checkpoint file to examine. The file name must include the remote node name.

DELETE REMOTECHKPT

`DELETE REMOTECHKPT` deletes a remote checkpoint reference.

Syntax

```
DELETE REMOTECHKPT checkpoint_file
```

checkpoint_file
The checkpoint file to examine.

INFO REMOTECHKPT

`INFO REMOTECHKPT` lists all remote checkpoint references.

Syntax

```
INFO REMOTECHKPT checkpoint_file
```

checkpoint_file
The checkpoint file to examine. You can specify a wildcard, as in `INFO REMOTECHKPT *`.

TMF commands

You can issue TMF commands for managing TMF dump information.

REFRESHTMFINFO

Forces the TMF information to be refreshed.

Syntax

```
REFRESHTMFINFO
```

TMFDUMPAGE

Specifies the number of days during which information will be returned for dumps that are created. The information returned by `TMFDUMPAGE` is limited to dumps that are created during the specified number of days.

Syntax

```
TMFDUMPAGE num_days
```

num_days
The number of days for limiting TMF dump information. The default is 30 days.

TMFDUMPINFO

`TMFDUMPINFO` returns information about TMF dumps on the local system.

Syntax

```
TMFDUMPINFO
```

TMFDUMPTABLEENTRIES

`TMFDUMPTABLEENTRIES` limits information returned by `TMFDUMPINFO` to the number of specified dumps.

Syntax

```
TMFDUMPTABLEENTRIES max_dumps
```

max_dumps
The number of dumps to display. The maximum allowed is 6000. The default is 1024 entries.

TMFREFRESHINTERVAL

Sets the refresh interval in seconds and writes reports to `ENV`. When you set a refresh interval for GGSCI, it overrides the `TMFREFRESHINTERVAL` that may have been specified in

the GLOBALS, Manager, or Extract parameter files. The override remains in effect for the duration of the current GGSCI session.

Syntax

```
TMFREFRESHINTERVAL seconds
```

seconds
The refresh interval in seconds. The default is 15 minutes or the value set in the GLOBALS parameter file.

TMFTRAILINFO

Use TMFTRAILINFO for diagnostic and informational purposes. Running TMFTRAILINFO will print information retrieved about the audit trails, such as the current active trail name and its enabled options.

Syntax

```
TMFTRAILINFO
```

Example

Sample output from a TMFTRAILINFO command:

```
\PROD.$AUDIT.ZTMFAT.AA
MinFiles 2, MaxFiles 5, Auditdump On
Active Vols $AUDIT
Restore Vols $DATA11
\MASTER.$DATA11.ZTMFAT.BB
MinFiles 2, MaxFiles 5, Auditdump On
Active Vols $DATA11
Restore Vols $AUDIT
```

Coordinator commands

Use Coordinator commands to establish a Coordinator group to monitor distributed network transactions. COORD is an alias for COORDINATOR in these commands.

ADD COORDINATOR

Use ADD COORDINATOR to add the process that will communicate with each node's Reader and Replicat processes to coordinate the application of distributed network transactions.

Syntax

```
ADD COORDINATOR group_name
[, CPU primary_cpu]
[, BACKUPCPU cpu]
[, PRI priority]
[, PROCESS process_name]
[, PARAMS param_file_name]
[, PROGRAM program_name]
[, REPORT report_name]
[, DESC "text"]
```

group_name
The group name.

CPU *primary_cpu*
The primary CPU on which Coordinator runs. The default is the CPU on which Manager runs.

BACKUPCPU *cpu*
An alternative CPU on which Coordinator runs if the primary CPU becomes unavailable.

PRI *priority*
The NonStop priority for the process. This defaults to the NonStop priority assigned to the TACL process underlying the `ADD`.

PROCESS *process_name*
The default process name is `$GGCnn`, where `nn` represents the sequence of the process. Oracle GoldenGate recommends that you use the default, however, if you must specify an alternative process, you can do so with the `PROCESS` *`process_name`* option.

PARAMS *param_file_name*
Specifies the alternative parameter file name to be used. Enter the fully qualified path name for the parameter file.

PROGRAM *program_name*
Specifies the name of the program that Manager assigns when starting the process. Typically this is not entered, and Manager uses the default `$GGCnn` name. The `HOST` parameter in the `GLOBALS` file determines the location of the default program.

REPORT *report_name*
Supplies an alternative report file name. The default report file name is *`install_volume`*`.GGSRPT.`*`rpt_nam`*`e`, where *`rpt_name`* represents the group name, such as `FINANCE`. Oracle GoldenGate creates an entry-sequenced file to hold each group's run results, and by default, the report name is the same as the group name.

DESC "*text*"
Describes the Coordinator group.

Example

```
ADD COORDINATOR TRXCO, CPU 2, PRI 150, DESC "Network transaction coordinator for NY,
FL, and LA"
```

ALTER COORDINATOR

Use `ALTER COORDINATOR` to change the checkpoints for an or to change the properties of an existing Coordinator group. You can use `ALTER COORDINATOR` to change the attribute of any option that you specified with `ADD COORDINATOR`.

Syntax

```
ALTER COORDINATOR group_name
[ trail_name
{BEGIN time |, EXTSEQNO seq_number, EXTRBA rba}]
[, options]
```

group_name
The group name.

| SOURCE {BEGIN *time* | , EXTSEQNO *seq_number* , EXTRBA *rba*}
Specifies the starting point in the Oracle GoldenGate trail as a beginning time, transaction sequence, or relative byte address. The specified must match one of the trails defined in the Coordinator parameter file.

options
The `ADD COODINATOR` options can be altered with this command. See `ADD COORDINATOR` for details.

Example

```
ALTER COORDINATOR TRXCO, CPU 1, PRI 180
```

DELETE COORDINATOR

Use `DELETE COORDINATOR` to remove a stopped Coordinator process from the system. `DELETE COORDINATOR` *`group_name`* removes the group and all checkpoints. Using the *`trail_name`* option deletes only the trail checkpoints, not the group.

Syntax

```
DELETE COORDINATOR group_name
[ trail_name]
```

group_name
The group name. Using this option without the deletes the group and all trail checkpoints.

trail_name
Deletes only the checkpoint.

Example

```
DELETE COORDINATOR TRXCO
```

INFO COORDINATOR

Use `INFO COORDINATOR` to display information on the attributes of the Coordinator.

Syntax

```
INFO COORDINATOR group_name
[, DETAIL]
[, SHOWCH]
[, PROGRAM]
```

group_name
The group name.

DETAIL
Reports Coordinator process run history, which includes starting and stopping points within the trail expressed as a time and the process parameters established by the `ADD COORDINATOR` command.

The default is to report the status of the Coordinator process (`STARTING`, `RUNNING`, `STOPPED` or `ABENDED`).

`SHOWCH`
Shows detailed historical checkpoints.

`PROGRAM`
Displays the name and location of the object that is running.

Example

The following is displayed from the command `INFO COORD TRXCO`

```
Coord   TRXCO         Last Started 2010-12-01 15:59  Status RUNNING
Process  $GGC00       Checkpoint Lag: unknown
Checkpoints:
    Trail                 Time                          Seqno         RBA
\NY.$DATA1.GGSDAT.Z1    Updated at 2010-12-01 16:00:22.950722
                          2010-11-17 12:22:46.657637       0           0
                          2010-11-17 12:22:46.657637       0        1779
\LA.$DATA1.GGSDAT.EX    Updated at 2010-12-01 16:00:22.950722
                          2010-12-01 15:55:39.664490       0           0
                          2010-12-01 16:00:11.437578       3   148578373
```

SEND COORDINATOR

Use `SEND COORDINATOR` to send a command to a running Coordinator process. Using `SEND COORDINATOR` you can perform the operations summarized in the table below.

Syntax

```
SEND COORDINATOR group_name {
GETREADERINFO |
GETTRANSINFO |
FORCECOMMIT transaction_id |
STATUS |
STOP}
```

`group_name`
The group name.

`GETREADERINFO`
Displays information about the Reader processes and the trails being read.

`GETTRANSINFO`
Displays information on pending transactions.

`FORCECOMMIT transaction_id`
Allows the transaction to be committed even though not all required trails have received the entire transaction. If a network connection is lost, for example, the parts of the transaction that are available can be committed.
Note: The ramifications of committing the partial transaction should be considered carefully before using `FORCECOMMIT`.

`STATUS`
Displays the current status of the Coordinator process.

STOP
Terminates the run gracefully. This command is preferable to stopping from TACL, which results in an ABEND status.

Examples

Example 1
An example of a display from SEND COORD TRXCO GETTRANSINFO:

```
  279: 0 TransID 7926335489872297987  2010/11/17 12:22:47.068460
     Pending, Needed 2, Found 1, RefCount 1
     Bitmap 8000 0000 0000
  813: 0 TransID 7926335489872232451  2010/11/17 12:22:46.947382
     Pending, Needed 2, Found 1, RefCount 1
     Bitmap 8000 0000 0000
  825: 0 TransID 7926335489872363523  2010/11/17 12:22:47.317281
     Pending, Needed 2, Found 1, RefCount 1
     Bitmap 8000 0000 0000
  909: 0 TransID 7926335489872166915  2010/11/17 12:22:46.769463
     Pending, Needed 2, Found 1, RefCount 1
     Bitmap 8000 0000 0000
 1701: 0 TransID 7926335489872101379  2010/11/17 12:22:46.657637
     Pending, Needed 2, Found 1, RefCount 1
     Bitmap 8000 0000 0000
```

Example 2
An example of a display from SEND COORD TRXCO GETREADERINFO:

```
Reader Information
 0 : \NY.$DATA1.GGSDAT.Z1, \NY.$ZRDR1, Node 109, POS 0,1779
     FastReads  Retries 0
     Current Transactions 5
     Oldest 7926335489872101379  2010/11/17 12:22:46.657637  0,1779
     Newest 7926335489872363523  2010/11/17 12:22:47.317281  0,4280
CurTransCount      5, LastRec 2010/12/01 16:01:34.104281
 Records           14, Bytes          924, Transactions           5
 1 : \LA.$DATA1.GGSDAT.EX, \LA.$ZRDR2, Node 109, POS 4,218437395
     FastReads  Retries 0
     No Current Transactions
CurTransCount      0, LastRec 2010/12/01 16:01:36.233081
 Records           11, Bytes          704, Transactions           0
Totals
Reader Requests           21, Records           25
Commit Requests            0
Force Commit 0
```

Example 3
An example of a display from SEND COORD TRXCO FORCECOMMIT 7926335489872297987:

```
TransID '7926335489872297987' set committable
```

START COORDINATOR

Use START COORDINATOR to start the Coordinator process. GGSCI routes the START request to Manager to start and monitor the process.

START COORDINATOR uses the READER option in Coordinator parameter file to identify the Reader processes that must be started and the trails that will be monitored. The following is an example of such a file.

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

In this example, starting the `TRXCO` Coordinator will start three Reader processes, each monitoring a trail on one of the three nodes, `\NY`, `\LA`, and `\FL`.

Syntax

```
START COORDINATOR group_name
```

group_name
The group name. You can use wildcards to specify a set of group names, such as, `*` or `TRX*`.

Example

```
START COORD TRXCO
```

STATUS COORDINATOR

Use `STATUS COORDINATOR` to determine if the Coordinator is running. A report displays to the Coordinator process's home terminal.

Syntax

```
STATUS COORDINATOR group_name [, DETAIL]
```

group_name
The group name. You can use wildcards to specify a set of group names, such as, `*` or `TRX*`.

`DETAIL`
When you specify `DETAIL`, (`STATUS COORD *, DETAIL`) checkpoint details are displayed. The default is to display the *`group_name`*, the status of the process and the process name.

Example

The following is an example display resulting from the command `STATUS COORD TRXCO`.

```
COORD    TRXCO      RUNNING (\NY.$GGC00)    ( 0,616 )  (140)
```

STOP COORDINATOR

Use `STOP COORDINATOR` to stop the Coordinator process gracefully.

Syntax

```
STOP COORDINATOR group_name
```

group_name
The group name.

Process commands

Process commands communicate using the process name. This is useful for tasks, such as a one-time data synchronization or direct file extraction, that are set up as special runs by using the SPECIALRUN, SOUCEISFILE, or SOURCEISTABLE parameters. These processes do not require an Extract or Replicat group name and can be identified only by the process name.

SEND PROCESS

Use SEND PROCESS to communicate with a running process using the process name rather than the group name. Once the process is started you can:

- Send commands recognized by the process
- Send a WAKE or BREAK command

Syntax

```
SEND PROCESS process_name {text | WAKE | BREAK}
```

process_name
The process name in the format *$identifier*.

text
One of the subset of GGSCI commands that will be recognized by the process. If there is a response from the process it will be displayed by GGSCI.

WAKE
Sends the WAKE command to the *process_name*.

BREAK
Sends the BREAK command to the *process_name.*

Example

The following example sends the STATUS command to the running process $GG12.

```
SEND PROCESS $GG12 STATUS
```

GGSCI will display the response, such as the following process status:

```
CUSTOM: Current Status: Waiting for more audit (seqno 360, rba 1208741308)
Audit Trail position: Seqno 360, Rba 1208741308
```

Marker commands

Markers are records inserted into the audit trails and log trails to identify application-specific events during Extract and Replicat processing.

For example, to switch from a primary to a backup database, you must determine that all records have been delivered from the primary to the backup database before switching. Markers provide a method for determining this without shutting down all TMF-related activity on the source node.

To determine that all records have been delivered, perform the following tasks:

1. Shut down application activity against the source database (for example, bring down PATHWAY).
2. Add a marker to the audit trail.
3. Wait until the corresponding Replicat process writes an event message indicating that it processed the marker. At this point, Replicat has processed all data from the source database and you can safely switch to the backup database.

Event messages and history records are written each time a marker is processed by GGSCI, Extract, or Replicat.

ADD MARKER

You can add markers:

- For TMF installations
- For non-TMF installations
- Executing TACL commands
- For a Replicat group

Default

By default, Oracle GoldenGate applies markers to the TMF audit trail.

Syntax

```
ADD MARKER
[LOGGER logger_prefix] [freeform_text] |
[TACLCMD program group_name command] |
[GROUPCMD] program group_name command ]
```

LOGGER *logger_prefix freeform_text*
See "Adding markers for TMF or non-TMF installations" for information on adding markers for TMF and non-TMF installations

TACLCMD *program group_name command*
See "Invoking TACL commands" for information on executing TACL commands.

GROUPCMD *program group_name command*
See "Adding markers for an Extract or Replicat group".

Adding markers for TMF or non-TMF installations

For TMF installations, ADD MARKER creates a marker in the local audit trail. For non-TMF installations, specify ADD MARKER with the LOGGER option.

Default

By default, Oracle GoldenGate applies markers to the TMF audit trail.

Syntax

```
ADD MARKER [LOGGER logger_prefix] [freeform_text]
```

LOGGER *logger_prefix*
Required to send markers to Logger processes and associated log trails. Identifies the Logger process group or an individual Logger process within a group. For example, `ADD MARKER LOGGER $GGL` specifies the Logger group and sends a marker to all processes beginning with `$GGL`. The command: `ADD MARKER LOGGER $GGL01`, sends a marker to the logger `$GGL01`.

freeform_text
Text you want added to the marker record to distinguish the purpose of the marker, as in:

```
ADD MARKER BROUGHT DOWN FINANCE
ADD MARKER LOGGER $GGL END OF DAY 2010-07-30
```

Invoking TACL commands

A special form of marker invokes TACL commands through Extract or Replicat. This lets you fit TACL commands into a stream of database activity. The command is carried out when Extract or Replicat encounters the marker record in the data stream.

Extract or Replicat end abnormally if encountering a problem issuing the command, but will not `ABEND` if the command itself fails. While the command runs, Extract or Replicat waits until it finishes. Specify `NOWAIT` as part of the command to return control immediately.

Syntax

```
ADD MARKER [LOGGER logger_prefix]
TACLCMD program group_name command
```

LOGGER *logger_prefix*
Required to send markers to Logger processes and associated log trails. Identifies the Logger process group or an individual Logger process within a group. For example, `ADD MARKER LOGGER $GGL` specifies the Logger group and sends a marker to all processes beginning with `$GGL`. The command: `ADD MARKER LOGGER $GGL01`, sends a marker to the Logger `$GGL01`.

TACLCMD *program group_name command*
The `TACLCMD` keyword informs the process that a TACL command is to be carried out. `TACLCMD` must include the following:

program
Either `EXTRACT` or `REPLICAT`. This determines which program runs the command.

group_name
The group name to run the command. You can specify a wildcard.

command
The TACL command to invoke. This can be the name of a program, a macro or any command that can be executed from TACL.

Example

In this example, `TACLCMD` specifies that the command is to be invoked by Extract for the `FINANCE` group. The command is: `FUP PURGEDATA $DATA1.DAT.FILE1`

```
ADD MARKER TACLCMD EXTRACT FINANCE FUP PURGEDATA $DATA1.DAT.FILE1
```

Adding markers for an Extract or Replicat group

You can use markers to send a command to an Extract or Replicat group.

Syntax

```
ADD MARKER GROUPCMD program group_name command
```

GROUPCMD *program group_name command*
The GROUPCMD keyword informs the process that a group command is to be invoked.

program
Enter Extract or Replicat.

group_name
Enter the Extract or Replicat group name.

command
Enter the command. Currently, CLOSEFILES is the only command available for GROUPCMD. CLOSEFILES instructs Replicat to close all opens on Enscribe files and SQL tables. It instructs Extract to close opens from FETCHCOMPS and FETCHLASTIMAGE.

INFO MARKER

Use INFO MARKER to review recently processed markers. A record is displayed for each occasion on which GGSCI, Logger, Extract or Replicat processed the marker.

Syntax

```
INFO MARKER [COUNT num_items]
```

COUNT *num_items*
Specify COUNT to restrict the list to the most recent number of items, as in: INFO MARKER COUNT 2.

Information returned includes:

PROCESSED
The local time that a program processed the marker.

ADDED
The local time at which the marker was inserted into the audit trails or log trails.

DIFF
The time difference between PROCESSED and ADDED. DIFF can serve as an indicator of the lag between application, Extract, and Replicat activities.

PROG
The process that processed the marker, such as GGSCI, Logger, Extract or Replicat.

GROUP
The Extract or Replicat group or Logger process that processed the marker. N/A is displayed if GGSCI processed the marker.

NODE
The node at which the marker was inserted into the audit trails.

Optional text
The free text you entered in the ADD MARKER command.

Programs commands

The PROGRAMS commands allow you to bind the GGSLIB intercept library to application programs, link GGSDLL intercept library to application programs, and view information about programs that may or may not be bound with GGSLIB.

BIND PROGRAMS

BIND PROGRAMS binds the TNS version of the GGSLIB intercept library to application programs. You must bind the intercept library to capture non-audited database updates to Enscribe files.

After issuing the BIND PROGRAMS command, you are prompted for a list of files with which to bind GGSLIB. You can enter a wildcard or actual program name. Terminate the list with GO (or cancel with EXIT). GGSLIB becomes the Guardian user library for specified programs (through the BIND CHANGE LIBRARY command).

If a program references a user library, that library is added to the bind list and GGSLIB is physically bound to the user library module (through the BIND BUILD command). The calling program's link to the user library is unchanged.

Syntax

```
BIND PROGRAMS
[, AXCEL| NOAXCEL]
[, PARAMS param_file_name]
[, REPORT report_file]
[, GGSLIB library_filename]
[, ERRORS num_errors]
[, FORCEBIND]
[, NOLIBBIND]
[, CHANGELIB]
```

AXCEL|NOAXCEL
AXCEL causes code acceleration after binding with existing user libraries. This option has no effect unless a user library is bound to GGSLIB and can be bypassed with NOAXCEL. If you do not specify NOAXCEL, Oracle GoldenGate will run BIND PROGRAMS with the AXCEL option enabled.

PARAMS *param_file_name*
The file that contains the program names to bind, as an alternative to entering file names interactively.

REPORT *report_file*
A file name for the detailed report of activity caused by this command. The default is *install_volume*.GGSRPT.BIND. Previous versions of the report file are aged to BIND00, BIND01, and so on.

GGSLIB *library_filename*
Changes the name of the GGSLIB to bind with the application. The default is GGSLIB in the Oracle GoldenGate home subvolume.

ERRORS *num_errors*
The number of allowable errors encountered by the BIND process before quitting. Default is 5.

FORCEBIND
Forces programs to be rebound with the library, even if they are already bound. Use FORCEBIND, for example, when binding a new release of GGSLIB to the application.

NOLIBBIND
Bypasses binding of existing user libraries with GGSLIB (default is to bind).

CHANGELIB
Instructs the bind process to change libraries to GGSLIB. Use this when receiving a new release of GGSLIB or BASELIB.

> **Note:**
>
> If your application programs are Native, then you must use the LINK PROGRAMS command to bind the Native version of the intercept library to your programs.

INFO PROGRAMS

Use INFO PROGRAMS to retrieve information about programs that may or may not be bound with GGSLIB. Use this command to determine if non-audited data will be extracted on a program-by-program basis.

Each program's modification timestamp is reported, so you can determine when data extraction took effect.

Syntax

```
INFO PROGRAMS file_name [, BOUND | UNBOUND]
```

file_name
A single file name or a wildcard list of files.

BOUND | UNBOUND

- BOUND limits the output to programs that have GGSLIB bound with either the program or the program's user library.
- UNBOUND reports programs that are not bound.

LINK PROGRAMS

LINK PROGRAMS links the native intercept library, GGSDLL, to your application programs. You must bind the intercept library to capture non-audited database updates to Enscribe files. Once this is complete, GGSDLL becomes the Guardian user library for specified programs (through the NLD -change libname command).

Syntax

```
LINK PROGRAMS
[, PARAMS param_file_name]
[, REPORT report_file]
[, GGSDLL library_filename]
[, ERRORS num_errors]
[, CHANGELIB]
```

After issuing the `LINK PROGRAMS` command, you are prompted for a list of files with which to link the library. You can enter a wildcard or actual program name. Terminate the list with `GO` (or cancel with `EXIT`).

`PARAMS param_file_name`
The file that contains the program names to link, as an alternative to entering file names interactively.

`REPORT report_file`
A file name for the detailed report of activity caused by this command. The default is `install_volume.GGSRPT.LINK`.
Previous versions of the report file are aged to `LINK00`, `LINK01`, and so on.

`GGSDLL library_filename`
Changes the name of the library to link with the application. The default name is `GGSDLL` for the operating systems. This library is stored in the Oracle GoldenGate installation subvolume.

`ERRORS num_errors`
The number of allowable errors encountered by the `NLD` process before quitting. Default is 5.

`CHANGELIB`
Instructs the link process to change libraries to `GGSDLL`. Use this when receiving a new release of BASELIBR and using NLDLIB.

Report commands

Extract, Replicat, Logger and Syncfile create reports about group process parameters, run statistics, error messages, and other diagnostic information. These reports can be created with the `SEND REPORT` command and viewed with the `VIEW REPORT` command. The report is displayed to your screen. Use the scrolling commands below to scroll through the report.

`return`
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 backward one page in file

q
Quit display

h
Help

SEND REPORT

Use `SEND REPORT` when you want to narrow reporting to a specific span of time, or to retrieve statistics about the current transaction.

Syntax

```
SEND [EXTRACT | REPLICAT | SYNCFILE] group_name,
REPORT [time_option [RESET | FILE name | TABLE name]]
```

group_name
The group name you defined with the appropriate GGSCI `ADD` command

`REPORT` *`time_option`* [`RESET` | `FILE` *`name`* | `TABLE` *`name`*
Returns Extract and Replicat processing statistics based on the selected options.

`RESET`
This resets the *`time_option`* counters to zero. For example, the command `SEND RECENT RESET` will report the `RECENT` counters and then reset them to zero.

`FILE` *`name`* | `TABLE` *`name`*
This limits the display to statistics for the file or table specified in *`name`*.

`time_option`
Each of these reports statistics for a different time interval as listed below. `REPORT` is the default. `TRANSACTION` is valid only for Replicat.
`REPORT`: since the last time the `REPORT` was run
`TOTALS`: since the Extract or Replicat was started
`DAILY`: since the beginning of the current day
`HOURLY`: since the beginning of the current hour
`RECENT`: since the last time the `RECENT` counter was reset
`TRANSACTION`: since the beginning of the current transaction

Examples

Example 1
The following sample Extract report uses the default `REPORT` option.

```
Report at 2010-02-27 12:43:07 (activity since 2010-02-27 11:43:44)
Elapsed time 0-00:59:22.916658
Total # records written to \NODEA.$TEST04.GGSDAT.ET               19
   \NODEA.$TEST04.GGSSOU.TCUSTMER           #  inserts:            3
                                            #  updates:            0
                                            #  deletes:            0
   \NODEA.$TEST04.GGSSOU.TCUSTORD           #  inserts:            3
                                            #  updates:           13
                                            #  deletes:            0
```

Example 2
The next sample Extract report uses the `TOTALS` option.

```
Report at 2010-02-27 12:44:15 counters for TOTALS since 2010-02-24 10:23:34
Elapsed time 3-02:20:41.216310
Total # records written to \NODEA.$TEST04.GGSDAT.ET               68
   \NODEA.$TEST04.GGSSOU.TCUSTMER            #  inserts:          11
                                             #  updates:           3
                                             #  deletes:           0
   \NODEA.$TEST04.GGSSOU.TCUSTORD            #  inserts:          15
                                             #  updates:          34
                                             #  deletes:           5
```

Example 3
The following Replicat report uses the `TRANSACTION` option.

```
Report at 2010-02-28 06:51:15 counters for TRANS since 2010-02-14 13:46:54
Elapsed time 13-17:04:21.290242
From \A.$TEST04.GGSSOU.ECUSTMER to \A.$TEST04.GGSTAR.HCUSTMER:
            #  inserts:         3
            #  updates:         1
            #  deletes:         0
            # discards:         0
From \A.$TEST04.GGSSOU.ECUSTORD to \A.$TEST04.GGSTAR.HCUSTORD:
            #  inserts:         3
            #  updates:         3
            #  deletes:         2
            # discards:         0
```

VIEW REPORT

`VIEW REPORT` allows read-only viewing of reports. Reports are aged each time Extract or Replicat is started. For example, if the report file for an Extract group is normally `GGSRPT.EXTCUST`, where `EXTCUST` is the group name, the reports are aged to `GGSRPT.EXTCUST0`, `GGSRPT.EXTCUST1`, and so on. This lets you trace through previous runs for diagnostic information.

Syntax

```
VIEW REPORT group_name[n] | file_name
```

group_name
The group name.

n
The sequence number of an aged report.

file_name
The fully qualified report file name. This is used when the report is not in the default location.

Syncfile commands

Syncfile commands let you manage the Syncfile utility, which duplicates entire files on a scheduled basis.

Use the `VIEW REPORT` command to view the output of Syncfile.

ADD SYNCFILE

Use `ADD SYNCFILE` to define a Syncfile group before starting Syncfile.

Using `ADD SYNCFILE` options you can:

- Specify alternative names for the parameter file, the report file, or the Syncfile process
- Specify a primary and backup CPU and associated NonStop priority

Syntax

```
ADD SYNCFILE group_name
[, PARAMS param_file_name]
[, REPORT report_file]
[, PROGRAM program_name]
[, PROCESS process_name]
[, CPU primary_cpu]
[, BACKUPCPU cpu]
[, PRI priority]
[, DESC "text"]
```

group_name
A Syncfile group. You can use wildcards to specify a set of group names, such as `*` or `*FIN*`.

PARAMS *param_file_name*
The Syncfile parameter file name.

REPORT *report_file*
The name of the report file to which Syncfile writes messages.

PROCESS *process_name*
The process name. See "Specifying an Alternative Process".

PROGRAM *program_name*
The name of the object to be run. See "Executing user exits".

CPU *primary_cpu*
The primary CPU name.

BACKUPCPU *cpu*
The CPU to use in the event the primary CPU is not available when starting or restarting Syncfile.

PRIORITY *priority*
The NonStop operating system priority.

DESC "*text*"
A description of the Syncfile process.

ALTER SYNCFILE

Use `ALTER SYNCFILE` to change an existing Syncfile group. Generally, you will use `ALTER SYNCFILE` to change `CPU`, `BACKUPCPU` or `PRIORITY` options.

Syntax

```
ALTER SYNCFILE group_name [, option]
```

group_name
A Syncfile group. You can use wildcards to specify a set of group names, such as `*` or `*FIN*`.

option
`ALTER SYNCFILE` uses the same options as `ADD SYNCFILE`.

DELETE SYNCFILE

Use `DELETE SYNCFILE` to delete a Syncfile group.

Syntax

```
DELETE SYNCFILE group_name
```

group_name
A Syncfile group. You can use a wildcard in the name.

INFO SYNCFILE

Use `INFO SYNCFILE` to display information about one or more Syncfile processes, including whether the process is running, and the date of the last time started.

Syntax

```
INFO SYNCFILE group_name [, DETAIL]
```

group_name
A Syncfile group. You can use a wildcard in the name.

DETAIL
Provides information regarding parameter, report files, process names and other information.

KILL SYNCFILE

Use `KILL SYNCFILE` to force a Syncfile process to stop immediately. Try `STOP SYNCFILE` first because it also performs cleanup work. `KILL SYNCFILE` is preferable to stopping Syncfile from TACL, which can result in Manager automatically restarting the process.

Syntax

```
KILL SYNCFILE group_name
```

group_name
A Syncfile group. You can use a wildcard in the name.

START SYNCFILE

Use `START SYNCFILE` to start one or more Syncfile processes. Manager receives the `START` request and starts and monitors the Syncfile process.

Syntax

```
START SYNCFILE group_name
```

group_name
A Syncfile group. You can use a wildcard in the name.

STATUS SYNCFILE

Use `STATUS SYNCFILE` to display the status of the Syncfile group.

Syntax

```
STATUS SYNCFILE group_name
```

group_name
A Syncfile group. You can use a wildcard in the name.

STOP SYNCFILE

Use `STOP SYNCFILE` to stop one or more Syncfile processes gracefully. Using this command lets you make configuration changes without affecting the operation of future Syncfile runs, and ensures that Manager will not restart the process.

Syntax

```
STOP SYNCFILE group_name
```

group_name
A Syncfile group. You can use a wildcard in the name.

Miscellaneous commands

The following commands control various other aspects of Oracle GoldenGate.

CLEANUP NETWORKCHECKPOINTS

Deletes the network checkpoint records in `repctxt` that were created by replicats when replicating files or tables that are partitioned across nodes.

Syntax

```
CLEANUP NETWORKCHECKPOINTS
```

! command

Use the `!` command to run a previous GGSCI command without modifications. To modify a command before reexecuting it, use the `FC` command. To display a list of previous commands, use the `HISTORY` command.

Issuing the `!` command without arguments reexecutes the most recently used command. By using options, you can reexecute a specific command by specifying its line number or a text substring.

Syntax

```
! [number | -number | string]
```

number
Reexecutes the command on the specified GGSCI line. Each GGSCI command line is sequenced, beginning with 1 at the start of the session.

-number
Reexecutes the command issued *number* lines before the current line.

string
Reexecutes the last command that starts with the specified text string.

Examples

Example 1

```
! 9
```

Example 2

```
! -3
```

Example 3

```
! sta
```

ENV

Use ENV to return information about the current run-time environment. This is based on the current settings for the GLOBALS parameters.

Syntax

```
ENV
```

Example

The following display illustrates the information returned for the ENV request.

```
Version                       192.0.2.2 H06 2010/05/14
Prefix                        $GG
System                        \NY
Programs                      \NY.$DATA1.GGS1040
Params                        \NY.$DATA1.GGSPARM
Report                        \NY.$DATA1.GGSRPT
LogFileOpens                  1
Manager Mandatory             Yes
Hometerm messages             Yes
SWAPVOL                       $DATA1
Reply timeout                 3000
TMF Refresh Interval          900
Current defines
=GGS_AUDCFG             CLASS MAP, FILE \NY.$DATA1.GGS.HHCFG
=GGS_PREFIX             CLASS MAP, FILE \NY.$GG
=_DEFAULTS CLASS DEFAULTS, VOLUME \NY.$DATA1.GGS8040
```

FC

Use FC to edit a previously issued GGSCI command and then reexecute it. Previous commands are stored in the memory buffer and you can display them by issuing the HISTORY command. The FC command is the same as standard NonStop FIX command functionality.

Issuing FC without arguments retrieves the most recently used command. By using options, you can retrieve a specific command by specifying its line number or a text substring. Previous commands can only be edited for the current command-line session, because command history is not maintained from session to session.

Using the editor

The FC command displays the specified command and then opens an editor with a prompt containing a blank line starting with two dots. Use the space bar to position the cursor beneath the character in the displayed command where you want to begin editing, and then enter one of the following arguments. Arguments are not case-sensitive and can be combined.

Table 1-1 FC editor commands

Argument	Description

`i text`	Inserts text. For example:
	`GGSCI> fc 9`
	`GGSCI> send mgr`
	`GGSCI..  i childstatus`
	`GGSCI> send mgr childstatus`

`r text`	Replaces text. For example:
	`GGSCI> fc 9`
	`GGSCI> info mgr`
	`GGSCI..  rextract ggfin`
	`GGSCI> info extract ggfin`

`d`	Deletes a character. To delete multiple characters, enter a d for each one. For example:
	`GGSCI> fc 10`
	`GGSCI> info extract ggfin, detail`
	`GGSCI..  dddddddd`
	`GGSCI> info extract ggfin`

replacement_text	Replaces the displayed command with the text that you enter on a one-for-one basis. For example:
	`GGSCI> fc 10`
	`GGSCI> info mgr`
	`GGSCI..  extract ggfin`
	`GGSCI> info extract ggfin`

To execute the command, press **Enter** twice, once to indicate there are no more changes and once to issue the command. To cancel an edit, type a slash (/) twice.

Syntax

```
FC [number | -number | string]
```

number
Returns the command from the specified line. Each GGSCI command line is sequenced, beginning with 1 at the start of the session.

-number
Returns the command that was issued *number* lines before the current line.

string
Returns the last command that starts with the specified text string.

Examples

Example 1

```
FC 9
```

Example 2

```
FC -3
```

Example 3

```
FC sta
```

HELP

Use HELP to view information about specific commands. For example, HELP displays information about the ADD REPLICAT command.

For a summary page displaying all commands and objects, enter the single command: HELP.

Syntax

```
HELP command object
```

command object
The command you need help with, such as ADD REPLICAT.

HISTORY

Use HISTORY to view a list of the commands issued in GGSCI. Previous commands are stored in GGSCIHST, an edit file located on the NonStop user's saved volume. Command history from each session of GGSCI remains available until the data is manually deleted from this file.

> **Note:**
>
> To clear the history, you can use TACL> WHO to find the saved volume for the user and then edit the GGSCIHST stored there.

You can use the ! command or the FC command to reexecute a command in the list. This command is the same as standard NonStop HISTORY command functionality.

Syntax

```
HISTORY [number]
```

number
Returns the last *number* commands, where *number* is any positive number.

Example

```
HISTORY 7
```

The result of this command would be similar to:

```
1: start manager
2: status manager
3: info manager
4: send manager childstatus
5: start extract ggfin
6: info extract ggfin
7: history
```

INFO ALL

Use INFO ALL to display a summary of the status and lag, where relevant, for each Oracle GoldenGate process. This display includes information for Manager, Extract, Replicat, Logger, and Syncfile.

Use INFO ALL to show the status of all active tasks and processes in the system. Use the SYSTEM command to address multiple systems at the same time.

Syntax

```
INFO ALL, [DETAIL | TASKS | ALLPROCESSES]
```

DETAIL
Reports process run history.

TASKS | ALLPROCESSES
Shows either tasks or all processes that are running. Specify either TASKS or ALLPROCESSES.

LOG

Using the LOG command, you can save the results of your GGSCI session to an output file. Use LOG to identify where to direct the session output. You can direct output to an edit file, spooler file, or a process name. Any new output will be appended to the file if it already exists.

Syntax

```
LOG {file_name | process_name | spooler}
```

file_name
The name of the output file.

process_name
The name of the process.

spooler
The name of the spooler.

LOG STOP

Use the `LOG STOP` command to close the session output.

Syntax

```
LOG STOP
```

OBEY

Use `OBEY` to process a file that contains a list of Oracle GoldenGate commands. `OBEY` is useful for executing commands that are frequently used in sequence. This command provides standard NonStop `OBEY` file functionality.

Syntax

```
OBEY file_name
```

file_name
The name of the file containing the list of commands.

Example

```
OBEY $DATA01.GGSPARM.FINANCE
```

EDIT PARAMS

`EDIT PARAMS` launches an editor from GGSCI.

Syntax

```
EDIT PARAMS file_name
```

file_name
Specify a parameter file name. If the file name is unqualified, `EDIT` assumes the parameter file resides in the Oracle GoldenGate home volume: `install_volume.GGSPARM`.

VIEW PARAMS

`VIEW PARAMS` displays the named parameter file to your screen. Use the scrolling commands described in Table 1-2 to scroll through the file.

Syntax

```
VIEW PARAMS file_name
```

file_name
Specify a parameter file name. If the file name is unqualified, `VIEW` assumes the parameter file resides in the Oracle GoldenGate home volume: `install_volume.GGSPARM`.

Table 1-2 Scrolling commands

Command	Description
return	Go to the 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 backward one page in file.
q	Quit display.
h	Help.

SYSTEM

The `SYSTEM` command enables you to manage Oracle GoldenGate processes from a single point of control. Use `SYSTEM` to switch the reference point from the local system to one or more remote systems, and then back again when needed. After making another system current, you can execute any command, subject to security constraints specified in the `CMDSEC` file on the remote system.

To enable the `SYSTEM` command, enter a `HOST` specification in the `GLOBALS` parameter file for each remote system with which you intend to communicate. The `HOST` entry requires the NonStop system name and `GGSSUBVOL` specification. These parameters identify and start GGSCI sessions on the remote systems.

Syntax

```
SYSTEM {system [, system] | ALL}
```

system
The NonStop system to make current. Omitting a system name defaults to the local system.

ALL
Makes all systems current.

Example

If Extract is running on the local system and Replicat is running on the remote system, the following commands would obtain information on each running process.

1. Get information about all Extract processes at `\LA`:

   ```
   GGSCI (\LA) > INFO EXTRACT *
   ```

2. Switch to `\NY`:

   ```
   GGSCI (\LA) > SYSTEM \NY
   ```

3. Get information about Replicats at `\NY`:

   ```
   GGSCI (\NY) > INFO REPLICAT *
   ```

4. Switch back to `\LA` (omitting system name defaults to local system):

   ```
   GGSCI (\NY) > SYSTEM
   ```

Example

You can address multiple systems at the same time:

1. Make the current systems `\LA` and `\NY`:

   ```
   GGSCI (\NY) > SYSTEM \LA, \NY
   ```

2. Get information about Extract and Replicat processes on both systems:

   ```
   GGSCI (\LA, \NY) > INFO ER *
   ```

VIEW GGSEVT

This command enables you to scroll through the Oracle GoldenGate event file (`LOGGGS`). This file contains event timestamps, text, program names and processes in chronological sequence.The `LOGGGS` file also includes a history of commands entered through GGSCI. Although this information is also recorded in the HP NonStop Event Management System (EMS), using `VIEW GGSEVT` is sometimes more convenient.

Syntax

```
VIEW GGSEVT
[, ASC | DESC]
[, TIME timestamp]
[, program]
[, event] [, SEARCH string]
```

`ASC | DESC`
`ASC` sorts the log in ascending order by time. `DESC`, the default, sorts entries in descending order.

`TIME timestamp`
Provides a starting point to look for event records. *`timestamp`* is expressed as `yyyy-mm-dd hh:mi` (for example `2014-03-31 12:30`).

`program`
Specify a program name to filter for events related only to that program: `EXTRACT`, `REPLICAT`, `LOGGER`, `SYNCFILE`, `MANAGER`, or `GGSCI`.

`event`
Specify one of: `START`, `STOP` (includes `ABEND` events), or `ABEND`.

SEARCH *string*
Searches for *string* in the log message. *string* must not contain spaces or be enclosed in quotes. You can specify multiple SEARCH *string* entries. If any string is found, the record is displayed.

GGSCI will display the events and then prompt: *enter* or q?

enter
Enter additional options for another GGSEVT display.

q
Exit GGSEVT and return to GGSCI.

2
Oracle GoldenGate Parameters

Learn how to use Oracle GoldenGate parameters for configuring, running, and managing the Oracle GoldenGate processes.
These parameters are grouped based on the processing to which they apply-logging changes, extracting data, replicating data, managing processes, or globally across all.

Parameter Summaries

This section summarizes Oracle GoldenGate parameters for HP NonStop, based on their functionality. This is followed by an alphabetized reference of the parameters.

GLOBALS Parameters Summary

`GLOBALS` parameters set global Oracle GoldenGate values to help standardize configuration and to perform other miscellaneous tasks. Normally, you would set global parameters when you install Oracle GoldenGate. Once set, you rarely need to change them, however, some of the parameters set in `GLOBALS` can be overridden by other Oracle GoldenGate programs.

The `GLOBALS` parameter file is stored in the subvolume where Oracle GoldenGate is installed. Use the editor to enter or change parameters in the `GLOBALS` file. You cannot change this file's name.

Oracle GoldenGate recommends setting global parameters before starting your first process. If you update the `GLOBALS` parameter file after starting a process, you must stop, then restart the process for the changes to take affect.

Table 2-1 GLOBALS Parameters: All

Parameter	Description
ADD DEFINE	Defines non-default locations for certain Oracle GoldenGate components.
AUDITING	Tells various programs in the system whether the installation is TMF, non-TMF, or both.
COMMENT	Indicates comments.
EMSLOG	Establishes a global EMS Collector name.
ERREPLYTIMEOUT	Sets the timeout value for GGSCI communication with Oracle GoldenGate components.
HEARTBEAT	Ensures that Extract and Logger can send data to Replicat.
HOMETERM	Sets up a global home terminal for Oracle GoldenGate processes.

Table 2-1 (Cont.) GLOBALS Parameters: All

Parameter	Description
HOMETERMMESSAGES	Suppresses or displays messages to the home terminal.
HOST	Identifies various NonStop nodes in the network.
IGNOREPARAMERROR	Prevents programs from terminating abnormally (abending) when an unrecognized `GLOBALS` entry is encountered.
IGNORETMFDUMPS	Prevents Extract, Manager, and GGSCI from capturing information about TMF disk and tape dumps.
LOGGERFLUSHRECS	Determines how many records Logger buffers before flushing to the log trail.
LOGFILEOPENS	Determines how many opens Logger keeps on the log files.
LOGGERFLUSHSECS | LOGGERFLUSHCSECS	Determines how many seconds Logger buffers records before flushing to the log trail.
LOGGGSCICOMMANDS	Determines whether user commands are written to the `LOGGGS` file.
MANAGERREQUIRED	Determines whether Extract and Replicat can run without a Manager process.
MAXWILDCARDENTRIES	Sets the initial allocation for wildcard entries.
OLDGROUPNAMING	Determines whether new Extract and Replicat group naming is enforced.
PARAMCHECK	Specifies whether Extract and Replicat check to ensure that the correct parameter file and process name have been specified at startup.
REPORTFILEEXTENTS	Allows the user to change the default `REPORTFILEEXTENTS` setting.
SUPPRESSMARKERMESSAGES	Suppresses messages generated when markers are processed by Extract, Replicat, or GGSCI.
SWAPVOL	Designates a swap volume for Oracle GoldenGate processes.
TMFDUMPAGE	Limits information that is returned when accessing the TMF catalog to dumps created during the specified number of days.
TMFDUMPTABLEENTRIES	Limits information that is returned when accessing the TMF catalog to the number of dumps specified.
TMFREFRESHINTERVAL	Sets the refresh interval.

Manager Parameters Summary

Manager parameters control how Manager interacts with different Oracle GoldenGate components such as Extract, Collector, and Replicat. GGSCI expects a parameter file named `MGRPARM` in the `GGSPARM` (or `prefixSPARM`) subvolume on the Oracle GoldenGate install volume. This name cannot be changed.

Table 2-2 Manager Parameters: General

Parameter	Description
BACKUPCPU	Sets the backup CPU for the Manager process.
CLEANUPSAVECOUNT	Changes the number of Extract and Replicat processing history records that are returned by the GGSCI `INFO EXTRACT` and `INFO REPLICAT DETAIL` commands.
COMMENT	Starts a comment line in the parameter file.
TMFDUMPAGE	Limits the amount of dump information returned from TMF dump process.
TMFDUMPTABLEENTRIES	Lets you store up to 6,000 TMF dump information entries.

Table 2-3 Manager Parameters: Process Management

Parameter	Description
AUTORESTART	Specifies processes to be restarted by Manager after a failure.
AUTOSTART	Specifies processes that are to be automatically started when Manager starts.
EXCLUDESUFFIXCHAR	Specifies characters that are not to be used as the first character of the suffix when generating process names.
MAXABENDRESTARTS	Determines how many times in `RESTARTINTERVAL` minutes Manager attempts to restart an abnormally terminated Extract, Replicat or Logger process.
RESTARTINTERVAL	Determines the period in which restart retries are counted before being reset.

Table 2-4 Manager Parameters: Maintenance

Parameter	Description
CHECKMINUTES	Determines how often Manager cycles through its maintenance activities.

Table 2-4 (Cont.) Manager Parameters: Maintenance

Parameter	Description
PURGEOLDEXTRACTS for Manager	Purges Oracle GoldenGate trails that are no longer needed according to specified criteria.
PURGEOLDTASKS	Purges Extract and Replicat tasks after a specified period of time.

Table 2-5 Manager Parameters: Port Management

Parameter	Description
DYNAMICPORTLIST	Specifies a range of ports that Manager can dynamically allocate.
IPINTERFACE	Restricts the Manager process to the interface specified by an IP address or DNS name.
PORT	Establishes the TCP/IP port number on which Manager listens for requests.
TCPIPPROCESSNAME	Specifies a TCP/IP process over which Manager listens for remote requests.

Table 2-6 Manager Parameters: Error and Event Management

Parameter	Description
DISKTHRESHOLD | NODISKTHRESHOLD	Generates an event message when an audit trail file that must be processed is in danger of being recycled.
DOWNCRITICAL	Includes a process that has terminated normally in the report generated by `DOWNREPORT`.
DOWNREPORT	Specifies how often down processes are reported to EMS.
LAGCRITICAL	Specifies a time lag for Extract and Replicat that is reported to EMS as a critical message.
LAGINFO	Specifies a time lag for Extract and Replicat that is reported to EMS as informational.
LAGREPORT	Sets an interval for reporting Extract and Replicat lags to EMS.
LOGFILESBEHIND | LOGFILESBEHINDINFO	Reports a critical or informational message when Extract or Replicat falls a certain number of files behind the current log trail file.
THRESHOLD | NOTHRESHOLD	Generates an event message when an audit trail file that needs to be processed is in danger of being recycled.
UPREPORT	Determines how often process "I'm alive" messages are reported.

Logger Parameters Summary

Logger performs data extracts when a NonStop source is non-TMF. By default, GGSCI expects the Logger parameter file to be named `LOGPARM` and located in *`install_volume`*`.GGSPARM.LOGPARM`. Enter parameters in the default file unless you have strong reasons to specify a different file name.

Table 2-7 Logger Parameters: General

Parameter	Description
COMMENT	Starts a comment line in the parameter file.
CPU	Specifies the primary and backup CPUs for the current logging process.
FORCESTOPDELAY	Causes the logging process to delay for a period of time before it is stopped by an operating system `STOP` or `ABEND`, or by a `STOP` or `ABEND` from an application.
HEARTBEAT	Causes Logger to issue heartbeat records every 60 seconds.
NOTSTOPPABLE	Enables Logger to be stopped only with the GGSCI `STOP LOGGER` command.
PRIORITY	Specifies the priority at which to run the current log process.

Table 2-8 Logger parameters: Trail and File Management

Parameter	Description
EXCLUDEFILE	Excludes a file or file set from extraction. Overrides all `FILE` settings for that Logger process.
FILE	Establishes a file or file set to be logged.
INCLUDESOURCEAPPINFO | EXCLUDESOURCEAPPINFO	Includes the source application program name and process ID as tokens in the trail.
LOGGERFLUSHRECS	Determines how many records Logger buffers before flushing to the log trail.
LOGGERFLUSHSECS | LOGGERFLUSHCSECS	Determines how many seconds Logger buffers records before flushing to the log trail.
LOG	Establishes a log process and the dimensions of the log trails for that process.
LOGFILEOPENS	Determines how many opens Logger keeps on the log files.
LOGGERFILENUM	Specifies the file number that `GGSLIB` uses to open the log process.

Table 2-8 (Cont.) Logger parameters: Trail and File Management

Parameter	Description
LOGGERTIMEOUTSECS	Controls how long GGSLIB will wait for a response from Logger before allowing the application to resume normal operations.

Table 2-9 Logger Parameters: Error and Event Management

Parameter	Description
DEBUGONSTACKCHECK	Instructs GGSLIB to call DEBUG whenever an application's process stack is close to overflowing.
RECEIVEQWARN	Issues an EMS warning if it receives a trail that exceeds the specified threshold.
TRACEALLOPENS	Instructs GGSLIB to send all open and close attempts on any file to Logger.
TRACECLOSES	Instructs GGSLIB to send close records to Logger for the purpose of tracing system activity.
TRACEOPENS	Instructs GGSLIB to send open records to Logger for the purpose of tracing system activity.
TRACEPROCESSIOS	Instructs Logger to precede each record logged with information regarding the process that created the record.
TRACESTATS	Instructs Logger to keep statistics for each process that sends it database operations.

CHGNOTE Parameters Summary

The CHGNOTE process is started whenever Logger is started, whenever a change is made to the Logger configuration, or whenever a FILE RENAME operation occurs. A separate CHGNOTE process is started in each CPU. Each process updates the last modified timestamp for each $SYSTEM.GGS.GGSCPU*nn* file, which notifies any user application that is bound to the Oracle GoldenGate Intercept library to reevaluate the Logger audit configuration segment.

By default, GGSCI expects the CHGNOTE parameter file to be named CHGPARM and located in the Oracle GoldenGate installation subvolume (not the GGSPARM subvolume). The CHGNOTE parameter file is optional.

Parameter	Description
RENAMEBUMPDELAY	Determines the number of seconds to bump the last modified timestamp on the GGSCPU*nn* files.

Extract Parameters Summary

Extract extracts source data from the TMF audit trail and writes it to one or more files, called Oracle GoldenGate trails. Extract parameter files are typically stored in the

GGSPARM subvolume with a file name that is the same as the group name. When using Extract checkpoints, the name of the Extract parameter file must match the entry designated in GGSCI for the Extract group.

Table 2-10 Extract Parameters: General

Parameter	Description
CHECKPARAMS	Verifies parameter file contents.
COMMENT	Indicates comments.
GETENV	Retrieves an environment variable set with the SETENV parameter.
INCLUDE	Identifies a macro library to be included.
OBEY	Accepts parameters from a different parameter file.
SETENV	Sets a NonStop environment variable.

Table 2-11 Extract parameters: Process Management

Parameter	Description
ALLOCFILES	Controls the amount of incremental memory that is allocated when the amount of memory specified with NUMFILES is reached.
ALTFILERESOLVE | NOALTFILERESOLVE	Resolves wildcards and identifies audited alternate key files on startup.
ALTINPUT	Distributes files across multiple Extract processes so that Extract never processes two files in sequence.
AUDSERVCACHEBLOCKS	Determines how much caching of SQL table definitions occurs when reading update records.
AUDSERVCPU	Begins reading the audit trail in a CPU different from that in which the Extract is running.
AUDSERVPARAM	Passes parameters that are specific to the Audserv process.
AUDSERVPREFIX	Permits Extract to assign custom prefixes to Audserv processes.
AUDSERVPROCESS	Permits Extract to assign an Audserv process a 5-character name.
AUDSERVPROGRAM	Allows customer to run both Native mode and TNS mode Audserv programs.
BEGIN	Establishes the time for Extract to start processing TMF audit trails. Required when SPECIALRUN is specified, otherwise omitted.
CHECKPOINTSECS	Specifies the maximum amount of time that Extract waits before writing a checkpoint.

Table 2-11 (Cont.) Extract parameters: Process Management

Parameter	Description
END	Establishes the END time for the current run. Not required unless SPECIALRUN is indicated. Online processing is implied if END is in the future or unspecified.
EXTRACT	Links this run to a particular Extract group. Required unless SPECIALRUN or SOURCEISFILE specified.
EXCLUDEGGSTRANSRECS | INCLUDEGGSTRANSRECS	Suppresses the creation of trail records that track distributed network transactions.
FILEAGEDAYS	Specifies the number of days a file can be inactive before it is aged off Extract's file list.
FILEEXCLUDE TABLEEXCLUDE	Excludes one or more files or tables from a file list.
FILERESOLVE	Controls whether wildcard lists are resolved during startup or dynamically.
FILTERVIEW | NOFILTERVIEW	Outputs a SQL view update only when a column has changed.
GETALTKEYS | IGNOREALTKEYS	Instructs Extract to produce (or ignore) file create records for alternate key files after it outputs the file create record for the primary file.
GETDEFAULTS	Lets you reset all Extract parameters to their default settings.
HEARTBEAT	Informs receiving processes (such as Replicat) that the source system is processing data.
LIMITRECS	Limits the number of records when capturing directly from a source table.
MAXWILDCARDENTRIES	Sets the initial allocation of wildcard entries.
NOFILEAGING	Deactivates the default of aging files from memory at two day intervals of no usage.
NUMFILES	Controls the initial allocation of memory dedicated to storing information about files to be processed by Oracle GoldenGate.
OPENTIMEOUT | OPENTIMEOUTMINUTES	Controls how long inactive files opened for FETCHCOMPS or FETCHLASTIMAGE remain open.
PASSTHRU | NOPASSTHRU	Allows Extract to distribute data from log trails when the source file has been renamed or purged.
SOURCEISFILE | SOURCEISTABLE	Reads all records from the source file rather than reading changes from the TMF audit trails. Useful for initial synchronization of source and target.
SPECIALRUN	Processes from user-specified BEGIN and END times. Used for one-time processing that does not require checkpointing from run-to-run. One of EXTRACT, SPECIALRUN or SOURCEISFILE is required.

Table 2-11 (Cont.) Extract parameters: Process Management

Parameter	Description
SYSKEYCONVERT	For direct file extraction only. Specifies the format of the syskey in the output file for entry-sequenced and ACI files.
TMFREFRESHINTERVAL	Sets the refresh interval in seconds.
VERBOSE	Allows Extract to allocate memory in large blocks.

Table 2-12 Extract Parameters: Security

Parameter	Description
DECRYPTTRAIL	Performs decryption when reading encrypted files in the Oracle GoldenGate trail.
ENCRYPTTRAIL | NOENCRYPTTRAIL	Encrypts data records in subsequent trail files until a `NOENCRYPT` is encountered. An Extract or Replicat process downstream must specify a `DECRYPT` to read the files.
LOGON	Specifies an alternate user ID under which Extract should run.

Table 2-13 Extract Parameters: File and Trail Management

Parameter	Description
EOFDELAY | EOFDELAYCSECS	Determines how many seconds Extract delays before looking for more data to extract when the source is an Oracle GoldenGate trail.
AUDITRETRYDELAY	Controls the frequency with which TMF audit trails are checked for new data.
DISPLAYTRAILSWITCH | NODISPLAYTRAILSWITCH	Causes Extract to write messages to the report file when it switches trails.
EXTFILE	Identifies the local Oracle GoldenGate trail as a flat file to which extracted data is output.
EXTTRAIL	Identifies the local Oracle GoldenGate trail to which extracted data is output.
FLUSHSECS | FLUSHCSECS	Determines the maximum time an extracted record remains buffered before being written to the Oracle GoldenGate trail.
NUMEXTRACTS	Allows Extract to exceed the default number of files in an Oracle GoldenGate trail.
OMITAUDITGAPCHECK	Enables Extract to continue processing even when a gap is detected between the oldest required audit trail and the current trail.

Table 2-13 (Cont.) Extract Parameters: File and Trail Management

Parameter	Description
PURGEOLDEXTRACTS for Extract and Replicat	Purges Oracle GoldenGate trail files when they are no longer required.
PURGERESTORE | NOPURGERESTORE	Purges audit files restored from tape immediately after use by Extract.
RESTORE | NORESTORE	Controls whether audit dumps on tape are reloaded when the dumps are not available on disk.
ROLLOVER	Specifies conditions under which local and remote Oracle GoldenGate trails start a new file in a sequence.
TMFTRAILTRACE	Instructs Extract to write messages to the report file when it switches to the next TMF audit trail.

Table 2-14 Extract Parameters: Remote Processing

Parameter	Description
IPINTERFACE	Restricts the Extract process to the interface specified by an IP address or DNS name.
POSITIONFIRSTRECORD	Positions Extract at the beginning of the input file to reprocess all records. For direct file extraction with `RMTBATCH` only.
RMTBATCH	Outputs data to a batch file on the target system. Enables activities such as trickle batch transfer.
RMTFILE	Identifies the remote Oracle GoldenGate trail as a single file.
RMTHOST	Establishes the remote TCP/IP host and port number to which output will be directed for subsequent `RMTFILE` and `RMTTRAIL` entries.
RMTHOSTALT	Specifies an alternative IP address in the event that the `RMTHOST` address is not reachable.
RMTTASK	Creates a task on the target system.
RMTTRAIL	Identifies the remote Oracle GoldenGate trail.
TCPBUFSIZE	Sets the maximum size for the message buffer sent to remote systems.
TCPFLUSHBYTES	Controls the TCP/IP flush size for better communications performance.
TCPIPPROCESSNAME	Changes the name of the TCP/IP process. Specifies additional processes as backups.
TCPIPSWITCHERRS	Determines how many retries on a TCP/IP connection are attempted before trying alternate addresses and controllers.

Table 2-14 (Cont.) Extract Parameters: Remote Processing

Parameter	Description
TCPSOURCETIMER | NOTCPSOURCETIMER	Adjusts timestamps of source records to synchronize with timestamp of target system.

Table 2-15 Extract Parameters: Performance Management

Parameter	Description
FASTPOSITION | NOFASTPOSITION	Instructs the AUDSERV program to do a binary search of the TMF Audit trail at startup, reducing startup time and CPU overhead associated with starting the process for the first time.
FASTIO	Writes output using large I/O transfers to increase performance.
FASTREADS | NOFASTREADS	Enables larger reads (up to 28K bytes) of trails.
FUNCTIONSTACKSIZE	Controls the size of the memory stack that is used for processing Oracle GoldenGate functions.
MAXTRANSMEM	Controls the maximum amount of memory allocated for a transaction.
SHORTREADDELAY	Optimizes system resource usage when Extract runs online by instructing Extract to delay a specified number of seconds whenever a block shorter than the optimal block size is read from a trail.

Table 2-16 Extract Parameters: Data Selection

Parameter	Description
COMPRESSDELETES | NOCOMPRESSDELETES	Extracts only the primary key fields or columns for deleted operations, which reduces network bandwidth and trail storage requirements.
FETCHCOMPS | FETCHLASTIMAGE	Controls how data from missing fields in a compressed update is supplied.
GETCOMPS | IGNORECOMPS	Includes or excludes compressed records in the extracted data.
GETCREATES | IGNORECREATES	Includes or excludes create records in the extracted data.
GETDELETES | IGNOREDELETES	Includes or excludes delete records in the extracted data.
GETFILEOPS | IGNOREFILEOPS	Includes or excludes file alter, create, purge, purgedata, rename and setmode records in the extracted data.

Table 2-16 (Cont.) Extract Parameters: Data Selection

Parameter	Description
GETINSERTS | IGNOREINSERTS	Includes or excludes insert records in the extracted data.
GETMARKERS | IGNOREMARKERS	Includes or excludes marker records in the extracted data.
GETNETCHANGES | IGNORENETCHANGES	Includes or excludes net update records in the extracted data.
GETNEWCOLUMNS | IGNORENEWCOLUMNS	Includes or excludes SQL column change records in the extracted data.
GETPARTONLYPURGEDATAS | IGNOREPARTONLYPURGEDATAS	Includes or excludes `PARTONLY PURGEDATA` operations in the extracted data.
GETPURGES | IGNOREPURGES	Includes or excludes file purge operations in the extracted data.
GETPURGEDATAS | IGNOREPURGEDATAS	Includes or excludes operations to purge file data in the extracted data.
GETRENAMES | IGNORERENAMES	Includes or excludes file rename records in the extracted data.
GETREPLICATES | IGNOREREPLICATES	Includes or excludes records produced by Replicat.
GETROLLBACKS | IGNOREROLLBACKS	Includes or excludes rollback records from being extracted.
GETUPDATEAFTERS | IGNOREUPDATEAFTERS	Includes or excludes update after-image records in the extracted data.
GETUPDATEBEFORES | IGNOREUPDATEBEFORES	Includes or excludes update before-image records in the extracted data.
GETUPDATES | IGNOREUPDATES	Includes or excludes update records in the extracted data.
GETAUXTRAILS | IGNOREAUXTRAILS	Ignores database changes stored in auxiliary TMF audit trails.
READTHRULOCKS | NOREADTHRULOCKS	Ignores record locks when performing `FETCHCOMPS`. This can help avoid a deadlock.

Table 2-17 Extract Parameters: Data Mapping

Parameter	Description
COLMATCH	Sets global column mapping rules.
FILE | TABLE	Outputs record changes from the named file into the current trail, subject to conditions and mapping specified as part of this parameter. At least one `FILE` or `TABLE` entry is required.

Table 2-18 Extract Parameters: Data Formatting and Conversion

Parameter	Description
CONVERTALLFLOATSTOIEEE | NOCONVERTALLFLOATSTOIEEE	Converts all float data type numbers in a table from Oracle GoldenGate HP NonStop Tandem (TDM) format to the Institute of Electrical and Electronics Engineers (IEEE) format.
CONVERTALLFLOATSTOTDM | NOCONVERTALLFLOATSTOTDM	Converts all float data type numbers in a table from IEEE format to TDM format.
FILLSHORTRECS | NOFILLSHORTRECS	Fills out records that are shorter than maximum length.
FORMATASCII | NOFORMATASCII	Formats output in external ASCII format.
FORMATSQL | NOFORMATSQL	Formats extracted records into equivalent `SQL INSERT`, `UPDATE`, and `DELETE` syntax.
FORMATXML	Formats extracted data into equivalent XML syntax.
FORMATLOCAL	Controls whether ASCII formatting occurs on NonStop or the target system.
REPLACEBADCHAR	Replaces invalid character values with specified entry.
REPLACEBADNUM	Replaces invalid numeric values with specified entry.
Y2KCENTURYADJUSTMENT | NOY2KCENTURYADJUSTMENT	Controls whether the Y2K window changes the century.

Table 2-19 Extract Parameters: Data Customization

Parameter	Description
COBOLUSEREXIT	Invokes COBOL85 user exit routines at different points during processing.
CUSEREXIT	Invokes C user exit routines at different points during processing.

Table 2-19 (Cont.) Extract Parameters: Data Customization

Parameter	Description
MACRO	Creates an Oracle GoldenGate macro.
MACROCHAR	Changes the macro character to something other than #.
EMBEDDEDMACROS | NOEMBEDDEDMACROS	Processes macros embedded within a quoted string.
TALUSEREXIT	Invokes TAL user exit routines at different points during processing.

Table 2-20 Extract Parameters: Reporting and Error Handling

Parameter	Description
DISCARDFILE	Specifies file name to contain records that could not be processed.
DISCARDROLLOVER	Specifies times at which to create new discard files.
DISPLAYFILEREFRESHES	Displays a message when file attributes are refreshed.
ERROR59ROLLOVER	Enables reading to skip ahead to the next file in a sequence upon encountering a damaged block (Guardian Error 59).
HOMETERMMESSAGES	Suppresses message display to the home terminal.
LAGSTATS	Specifies parameters for capturing lag and related statistics.
LIST | NOLIST	Control whether the macros of a macro library are listed in the report file.
REPORT	Schedules a statistical report at a specified date or time.
REPORTCOUNT	Reports records processed at defined intervals.
REPORTROLLOVER	Specifies times at which to create new report files.
RETRYERR	Sets up error retry options (for example, to retry network errors automatically).
STATOPTIONS	Optionally reports zero counts for insert, update and delete operations

Table 2-20 (Cont.) Extract Parameters: Reporting and Error Handling

Parameter	Description
SUPPRESSALLALTERMESSAGES | NOSUPPRESSALLALTERMESSAGES	Suppress all messages produced when Extract finds a `FILE ALTER` record in the TMF audit trail.
SUPPRESSALTERMESSAGES | NOSUPPRESSALTERMESSAGES	Suppress all except the first messages produced when Extract finds a `FILE ALTER` record in the TMF audit trail.
SUPPRESSFETCHCOMPRESSEDDISCARDS | NOSUPPRESSFETCHCOMPRESSEDDISCARDS	Suppress display of messages when a `FETCHCOMP` fails to find the needed record.
SUPPRESSFILEOPMESSAGES | NOSUPPRESSFILEOPMESSAGES	Suppress the output of messages generated after `FILE RENAME`, `PURGE`, `CREATE`, `ALTER`, `SETMODE`.
SUPPRESSMARKERMESSAGES	Suppress messages generated when markers are processed.
VERSIONERR	Specifies error handling when database definitions are out of synchronization with audit trail records.

Table 2-21 Extract Parameters: Application-Specific

Parameter	Description
DICTIONARY	Opens a DDL dictionary. Used to associate record layouts with Enscribe files.
EXPANDDDL	Enables custom specification of Enscribe array items and formatting of field names.

Replicat Parameters Summary

Replicat reads data from Oracle GoldenGate trails that were created by Extract or Logger. Replicat parameter files are typically stored in the `GGSPARM` subvolume with a file name that is the same as the group name. When using Replicat checkpoints, the name of the Replicat parameter file must match the entry designated in GGSCI for the Replicat group.

Table 2-22 Replicat Parameters: General

Parameter	Description
CHECKPARAMS	Verifies parameter file contents.
COMMENT	Indicates comments.

Table 2-22 (Cont.) Replicat Parameters: General

Parameter	Description
GETENV	Retrieves an environment variable set with the SETENV parameter.
INCLUDE	Identifies a macro library to be included.
OBEY	Accepts parameters from a different parameter file.
SETENV	Sets a NonStop environment variable.

Table 2-23 Replicat Parameters: Process Management

Parameter	Description
ALLOCFILES	Controls the amount of incremental memory that is allocated when the amount of memory specified with NUMFILES is reached.
ASSUMETARGETDEFS	Assumes that the source files or tables are structured like the target, bypassing the retrieval process from source system.
AUDITREPS | NOAUDITREPS	Turns auditing on or off for Replicat operations.
BEGIN	Establishes the time for Replicat to begin processing the Oracle GoldenGate trails. Required when SPECIALRUN specified, otherwise omitted.
CHECKPOINTSECS	Controls how often Replicat writes a checkpoint when checkpoints are not being generated as a result of transaction commits.
CHECKUNIQUEKEY | NOCHECKUIQUEKEY	Forces Replicat to check for unique key constraint violations on entry-sequenced files before inserting data.
COORDINATOR	Identifies the process that will coordinate transactions that are distributed across multiple nodes.
DICTIONARY	Opens a DDL dictionary. Used to associate definitions with Enscribe files for column mapping and selectivity condition evaluation.
END	Establishes the END time for the current run. Not required unless SPECIALRUN is specified. Online processing is implied if END is in the future or unspecified.
ENTRYSEQUPDATES	Applies changes to entry-sequenced files exactly as they were on the source system (non-audited files only).
EOFDELAY | EOFDELAYCSECS	Determines how many seconds Replicat delays before looking for more data to replicate.
EXPANDDDL	Enables custom specification of Enscribe array items and formatting of field names.

Table 2-23 (Cont.) Replicat Parameters: Process Management

Parameter	Description
EXTFILE	Provides the source of records to be replicated. Either `EXTFILE` or `EXTTRAIL` must be specified for `SPECIALRUN`.
EXTTRAIL	See `EXTFILE`.
FASTREADS | NOFASTREADS	Enables larger reads (up to 28K bytes) of Oracle GoldenGate trails.
FILEAGEDAYS	Specifies the number of days a file can be inactive before it is aged off Replicat's file list. See ticket 21181027.
FILEOPWARNING	Controls Replicat's behavior when it attempts to purge non-existent files.
FLUSHCHECKPOINT | NOFLUSHCHECKPOINT	Controls whether files are flushed from an Oracle GoldenGate trail when Replicat records a checkpoint.
FORCEUSESYSKEY | NOFORCEUSESYSKEY	Forces Replicat to specify the `SYSKEY` when executing updates and deletes on entry-sequenced and cluster-sequenced SQL tables. It forces the syskey to be -1 for relative Enscribe files.
GETDEFAULTS	Lets you reset all Extract parameters to their default settings.
GETNETWORKALTFILENAMES | IGNORENETWORKALTFILENAMES	Controls whether the file system qualifies the alternate key file names with the local node name.
GROUPTRANSOPS	Groups multiple transactions within larger transactions, which can greatly enhance performance.
MAXDISCARDRECS	Limits the number of discarded records reported to the discard file.
MAXETCHECKPOINTSECS	Specifies the maximum amount of time that Replicat waits before writing a checkpoint.
MAXWILDCARDENTRIES	Sets the initial allocation of wildcard entries.
NETWORKCHECKPOINTS	When Replicat encounters partitions residing on a remote node, triggers update of the local checkpoint file to allow identification of replicated data.
NOFILEAGING	Deactivates the default of aging files from memory at two day intervals of no usage.
NUMFILES	Controls the initial allocation of memory dedicated to storing information about files to be processed by Oracle GoldenGate.
OPENTIMEOUT | OPENTIMEOUTMINUTES	Controls how long inactive files opened for `FETCHCOMPS` or `FETCHLASTIMAGE` remain open.
PARTMAP	Specifies alternate partitioning schemes during file creation on the backup system.

Table 2-23 (Cont.) Replicat Parameters: Process Management

Parameter	Description
POSITIONFIRSTRECORD	Positions Replicat at the beginning of the input file to reprocess all records. For direct file extraction with RMTBATCH only.
PURGEDATAALTFILES | NOPURGEDATAALTFILE	Purges data on the alternate key files when an Enscribe PURGEDATA is received for the primary file.
PURGEOLDEXTRACTS for Extract and Replicat	Purges Oracle GoldenGate trails when they are no longer required.
REPLICAT	Links this run to a particular Replicat group (set up with GGSCI) to facilitate continuous processing through the Oracle GoldenGate trails. Required unless SPECIALRUN is entered.
REPNEWCOLUMNS | NOREPNEWCOLUMNS	Replicates SQL ALTER TABLE ADD COLUMN statements.
REVERSEWINDOWSECS | REVERSEWINDOWCSECS	Holds transactions for a specified number of seconds or centiseconds to wait for an out-of-order transaction to be processed. Valid for non-audited target files.
SOURCEDEFS	Specifies a text file that contains Oracle GoldenGate source file and table data definitions (created by the DEFGEN utility).
SPECIALRUN	Processes from user specified BEGIN and END times. Used for one-time processing that does not require continuous processing of Oracle GoldenGate trails from run to run. Either REPLICAT or SPECIALRUN is required.
WAITFILEEVENT	Waits for a file-related event to occur before proceeding.

Table 2-24 Replicat Parameters: Security

Parameter	Description
DECRYPTTRAIL	Use DECRYPTTRAIL to decrypt encrypted Oracle GoldenGate trail files.
LOGON	Specifies an alternate user ID under which to run Replicat.

Table 2-25 Replicat Parameters: Data Selection

Parameter	Description
GETCREATES | IGNORECREATES	Includes or excludes file create records from a particular trail.
GETDELETES | IGNOREDELETES	Includes or excludes delete records from being replicated.

Table 2-25 (Cont.) Replicat Parameters: Data Selection

Parameter	Description
GETFILEOPS | IGNOREFILEOPS	Includes or excludes file alter, create, purge, purgedata, rename and setmode records from being replicated.
GETINSERTS | IGNOREINSERTS	Includes or excludes insert records from being replicated.
GETNEWCOLUMNS | IGNORENEWCOLUMNS	Includes or excludes detection of new SQL columns.
GETPARTONLYPURGEDATAS | IGNOREPARTONLYPURGEDATAS	Includes or excludes `PARTONLY PURGEDATA` operations.
GETPURGES | IGNOREPURGES	Instructs Replicat to include or ignore the `FUP PURGE` operation.
GETPURGEDATAS | IGNOREPURGEDATAS	Instructs Replicat to include or ignore the `FUP PURGEDATA` operation.
GETRENAMES | IGNORERENAMES	Includes or excludes file rename records in the current Oracle GoldenGate trail.
GETUPDATEAFTERS | IGNOREUPDATEAFTERS	Includes or excludes update after-images from being replicated.
GETUPDATEBEFORES | IGNOREUPDATEBEFORES	Includes or excludes update before-images from being replicated.
GETUPDATES | IGNOREUPDATES	Includes or excludes update records from being replicated.
INSERTALLRECORDS | NOINSERTALLRECORDS	Causes Replicat to insert every change operation made to a record as a new record in the database.
INSERTMISSINGUPDATES | NOINSERTMISSINGUPDATES	Inserts update operations that have no corresponding records already inserted into the target file.
OVERRIDEDUPS | NOOVERRIDEDUPS	Overlays the current insert record onto an existing record whenever a duplicate record error occurs. Otherwise, an error results.
UPDATEDELETES | NOUPDATEDELETES	Changes deletes to update operations. Useful for archiving and reporting.

Table 2-26 Replicat Parameters: Data Mapping

Parameter	Description
COLMATCH	Sets global column mapping rules.

Table 2-26 (Cont.) Replicat Parameters: Data Mapping

Parameter	Description
MAP	Specifies a source to target file pair. At least one `MAP` entry is required.
MAPEXCLUDE	Excludes one or more files from a wildcard map list.

Table 2-27 Replicat Parameters: Data Formatting and Conversion

Parameter	Description
CONVERTALLFLOATSTOIEEE | NOCONVERTALLFLOATSTOIEEE	Converts all float data type numbers in a table from Oracle GoldenGate HP NonStop Tandem (TDM) format to the Institute of Electrical and Electronics Engineers (IEEE) format.
CONVERTALLFLOATSTOTDM | NOCONVERTALLFLOATSTOTDM	Converts all float data type numbers in a table from IEEE format to TDM format.
FILLSHORTRECS | NOFILLSHORTRECS	Fills out records that are shorter than maximum length.
INSERTDELETES | NOINSERTDELETES	Changes deletes to insert operations. Useful for archiving and reporting.
INSERTUPDATES | NOINSERTUPDATES	Changes updates to insert operations.
REPLACEBADCHAR	Replaces invalid character values with specified entry.
REPLACEBADNUM	Replaces invalid numeric values with specified entry.
UPDATEDELETES | NOUPDATEDELETES	Changes deletes to update operations. Useful for archiving and reporting.
UPDATEINSERTS | NOUPDATEINSERTS	Changes insert operations to update operations. Useful for archiving and reporting.
Y2KCENTURYADJUSTMENT | NOY2KCENTURYADJUSTMENT	Controls whether the Y2K window changes the century.

Table 2-28 Replicat Parameters: Data Customization

Parameter	Description
COBOLUSEREXIT	Instructs Replicat to invoke COBOL85 user exit routines at different points during processing.
CUSEREXIT	Instructs Replicat to invoke C user exit routines at different points during processing.
MACRO	Creates an Oracle GoldenGate macro.
MACROCHAR	Changes the macro character to something other than #.
EMBEDDEDMACROS | NOEMBEDDEDMACROS	Processes macros embedded in a quoted string.
TALUSEREXIT	Invokes TAL user exit routines at different points during processing.

Table 2-29 Replicat Parameters: Reporting and Error Handling

Parameter	Description
CONTROLTABLELOCKOFF	Used as a global level toggle before and after map statements as well as an option of a map.
DISCARDFILE	Contains records that could not be processed.
DISCARDROLLOVER	Specifies times at which to create new discard files.
DISPLAYFILEREFRESHES	Displays a message when file attributes are refreshed.
HANDLECOLLISIONS | NOHANDLECOLLISIONS	Ignores duplicate record and missing record errors. Useful for overlaying database changes that occur during initial loading and other purposes.
LAGSTATS	Specifies parameters for capturing lag and related statistics.
LIST | NOLIST	Controls whether the macros of a macro library are listed in the report file.
OPENWARNINGS	Determines how Replicat processes Enscribe file open warnings.
REPERROR	Determines how Replicat responds to certain Guardian and SQL errors, including how transactions are backed out.
REPORT	Schedules a statistical report at a specified date or time.
REPORTCOUNT	Reports records processed at defined intervals.

Table 2-29 (Cont.) Replicat Parameters: Reporting and Error Handling

Parameter	Description
REPORTROLLOVER	Specifies times at which to create new report files.
REPORTTMFEXCEPTIONS	Reports when Replicat detects a `TMFEXCEPTIONS` type of condition
REPSQLLOG	Redirects the log to a file other than the Replicat report file.
RESTARTCOLLISIONS | NORESTARTCOLLISIONS	Enables `HANDLECOLLISIONS` until the first checkpoint is finished.
RETRYDELAY	Specifies the delay between attempts at retrying a failed insert, update, or delete operation.
SQLFORMATDISCARDFILE	Contains SQLCI input for discarded SQL/MP operations
SQLFORMATDISCARDROLLOVER	Specifies when to create new SQL formatted discard files
STATOPTIONS	Optionally reports on time spent creating files, purging files, and ending transactions. Optionally reports zero counts for insert, update and delete operations.
SUPPRESSFILEOPMESSAGES | NOSUPPRESSFILEOPMESSAGES	Suppresses the output of messages generated after `FILE RENAME`, `PURGE`, `CREATE`, `ALTER`, `SETMODE`.
SUPPRESSMARKERMESSAGES	Suppresses messages generated when markers are processed.
TMFEXCEPTIONS | NOTMFEXCEPTIONS	Resolves certain out-of-order records in the TMF audit trail within a given transaction.
WARNRATE	Determines how often database errors are reported. Useful when many errors are anticipated; controls the size of the log.

Coordinator Parameters Summary

Coordinator tracks distributed network transactions to coordinate updates on multiple nodes.

Table 2-30 Coordinator Parameters: All

Parameter	Description
COORDINATOR	Links this run to a particular Coordinator group.
FASTREADS | NOFASTREADS	Enables larger reads (up to 28K bytes) of trails.

Table 2-30 (Cont.) Coordinator Parameters: All

Parameter	Description
READER	Triggers the creation of a Reader process to access a local trail and communicate with the system Coordinator

Syncfile Parameters Summary

Syncfile lets you schedule and manage file duplication when you wish to copy files in their entirety.

Table 2-31 Syncfile Parameters: All

Parameter	Description
ABORTDUPERRWINDOW	Specifies a time period for Syncfile to restart aborted processes before ending abnormally.
CHECKINTERVAL	Lets you change the time interval for evaluating scheduled events.
COMMENT	Starts a comment line in the parameter file.
DUP	Specifies options for duplication, including the file set to duplicate, the target file set, the duplication method and schedule. At least one DUP parameter is required.
DUPONLYAFTEREVENT	Specifies a time for events to execute.
DUPPROCESS	Lets you specify the process name used for the TACL or FUP process used to duplicate files. This parameter is useful for debugging.
EVENT	Identifies an event, by name, and supplies its schedule. At least one EVENT parameter is required.
HOMETERMMESSAGES	Suppresses message display to the home terminal.
REPORTROLLOVER	Specifies when to age the current report file and create a new one.
SYNCFILE	For use when the parameter file has a different name than the group name.
TRACE	Logs Syncfile processing messages to a report file.
VERBOSE	Turns on display of FUP and TACL messages.

ABORTDUPERRWINDOW

Valid for

Syncfile

Description

Use `ABORTDUPERRWINDOW` to specify an alternative time interval for Syncfile to restart processes that abend.

To duplicate files, Syncfile starts either a FUP or TACL process to perform the duplication. If one of these processes abends, Syncfile automatically restarts it. However, if the process abends more than three times within one hour, the Syncfile process abends.

Default

```
1 HOUR
```

Syntax

```
ABORTDUPERRWINDOW num_units unit;
```

num_units
The number of units of time, as in 1 or 10.

unit
The time unit. Specify one of: `SECONDS`, `MINUTES` or `HOURS`.

Example

```
ABORTDUPERRWINDOW 4 HOURS;
```

ADD DEFINE

Valid for

`GLOBALS`

Description

`ADD DEFINE` provides the ability to specify valid `DEFINE` statements to help with resolution of default locations, file names, or anything that can be set using a standard `DEFINE`.

Oracle GoldenGate recommends that you use the defaults, however, if your installation requires that you use other definitions, you can define them in the `GLOBALS` parameter file using `ADD DEFINE`. By specifying defines in the `GLOBALS` parameter file, they are established for all the affected components you install; you don't have to specify them in other parameter files.

Default

`ADD DEFINE` changes the following defaults:

- The default for the Oracle GoldenGate environment prefix, `GGS`.

- The default volume and subvolume for AUDCFG, $SYSTEM.GGS.AUDCFG.
- The default number of seconds the Logger process delays before closing the old file when rolling to a new file.
- The default volume and subvolume for parameter files, $VOL.GGSPARM.
- The default volume and subvolume for report files, $VOL.GGSRPT.

> **Note:**
>
> Defaults set by ADD DEFINE for =GGS_PARAMS or =GGS_REPORT can be overridden by the appropriate GGSCI ADD or ALTER command.

- The default TCPIP process name, $ZTC0

> **Note:**
>
> Defaults set by the ADD DEFINE for =TCPIP^PROCESS^NAME can be overridden by using the TCPIPPROCESSNAME parameter for Manager or Extract or using the RMTHOST TCPIPPROCESSNAME option.

When you create a new volume, subvolume, or file name, you need to specify the new location and file name in the appropriate places, such as when you execute a GGSCI ADD or ALTER command, or in any parameter file where you specified a location or file name.

Syntax

```
ADD DEFINE {define}, [class_node_vol_spec]
```

define
The *define* must be one of the following:

=GGS_AUDCFG
Changes the default file name of the AUDCFG file.

=GGS_DB_SUBVOL
Defines the subvolume location of the Oracle GoldenGate database.

=GGS_LOGFILE_CLOSE_DELAY
Changes the delay before Logger closes the old file when rolling to a new one. The default is 120 seconds.

=GGS_PARAMS
Changes the default subvolume used to store parameter files.

=GGS_PREFIX
Lets you define a different Oracle GoldenGate prefix for process names and subvolumes.

=GGS_REPORT
Changes the default subvolume used to store the report file.

=TCPIP^PROCESS^NAME
Changes the default TCPIP process name.

class_node_vol_spec
If you are defining a new volume and subvolume or a file name enter *class_node_vol_spec* as follows:

- To specify a file name, enter:

  ```
  ADD DEFINE =GGS_type, CLASS MAP,
  FILE \NODE.$volume.subvolume.file_name
  ```

- To specify a new default volume and subvolume, enter:

  ```
  ADD DEFINE =GGS_type, CLASS DEFAULTS,
  VOLUME \NODE.$volume.subvolume
  ```

- To specify a new prefix, enter:

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

- To specify a delay time for closing the log file, enter:

  ```
  ADD DEFINE =GGS_LOGFILE_CLOSE_DELAY, CLASS MAP, FILE $xseconds
  ```

  Where `FILE` is up to five digits for *seconds* and *x* can be any character a to z.

Examples

Example 1
This example changes the default subvolume used to store parameter files to `$GGSPROD`.

```
ADD DEFINE =GGS_PARAMS, CLASS DEFAULTS, VOLUME \PROD.$DATA3.GGSPROD
```

Example 2
This example changes the default subvolume used to store the report files.

```
ADD DEFINE =GGS_REPORT, CLASS DEFAULTS, VOLUME \PROD.$DATA3.NEWRPT
```

Example 3
This example changes the location of the audit configuration file.

```
ADD DEFINE =GGS_AUDCFG, CLASS MAP, FILE \NODE.$DATA1.GGS.AUDCFG
```

Example 4
This example identifies the subvolume where the default database is installed. This is not recommended because moving the database can cause problems when migrating to a new release of Oracle GoldenGate. If Extract, Replicat, or Syncfile are moved to a different subvolume it may be necessary to use the defines since they must know where to find checkpoint files.

```
ADD DEFINE =GGS_DB_SUBVOL, CLASS DEFAULTS, VOLUME, \PROD.$DATA6.GGS
```

Example 5
This example sets a delay of 240 seconds for Logger to wait before closing the old file when rolling to a new file.

```
ADD DEFINE =GGS_LOGFILE_CLOSE_DELAY, CLASS MAP, FILE $a240
```

Example 6
This example changes the default `$ZTC0` TCPIP process name to `$ZTC3`.

```
ADD DEFINE =TCPIP^PROCESS^NAME, FILE $ZTC3
```

> **Note:**
>
> For more information on working with defines, see Changing Default Component Names.

ALLOCFILES

Valid for

Extract, Replicat

Description

Use `ALLOCFILES` to control the incremental number of memory structures allocated once the initial memory allocation specified by the `NUMFILES` parameter is reached (see "NUMFILES"). Together, these parameters control how process memory is allocated for storing information about the source and target tables being processed.

The default values should be sufficient for both `NUMFILES` and `ALLOCFILES`, because memory is allocated by the process as needed, system resources permitting.

`ALLOCFILES` must occur before any `TABLE` or `MAP` entries to have any effect.

Default

500

Syntax

`ALLOCFILES` *`num_structures`*

`num_structures`
The additional number of memory structures to be allocated. Do not set `ALLOCFILES` to an arbitrarily high number, or memory will be consumed unnecessarily. The memory structures of Oracle GoldenGate support up to two million tables.

Example

```
ALLOCFILES 1000
```

ALTFILERESOLVE | NOALTFILERESOLVE

Valid for

Extract

Description

Use `ALTFILERESOLVE` to resolve wildcards during the startup of Extract. As the wildcards are resolved, audited alternate key files are placed on the "exclude list" so no data will be sent for them. This avoids duplicate key errors that can otherwise occur when

alternate key file I/O is encountered before primary file I/O. A message is produced for each wildcard entry as it is processed.

Unstructured files, SQL/MP tables, SQL/MX tables, and NSK standard subvolumes are ignored during the scan.

After startup, any file create or alter operation triggers Extract to evaluate the file for audited alternate key files to add to the exclude list.

To avoid the initial scan and its startup overhead, set `NOALTFILERESOLVE`. When this is set, alternate key files will be identified when the first primary file I/O is encountered.

Default

```
ALTFILERESOLVE
```

Syntax

```
ALTFILERESOLVE | NOALTFILERESOLVE
```

Example

The following is an example of the message produced during a wildcard scan at startup.

```
2010-02-16 08:10:58.161307 Scanning $*.A12GEN* for Alternate Key Files
2010-02-16 08:10:58.230067 Files scanned 16
2010-02-16 08:10:58.230788 Scanning $*.A12SQL* for Alternate Key Files
2010-02-16 08:10:58.289293 Files scanned 72
```

ALTINPUT

Valid for

Extract

Description

Use `ALTINPUT` for direct file extraction. With ACI files, multiple files can be in use at one time. For example, processing can continue on 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.

To set up Extract for direct file extraction, specify the `FILETYPE` option in the GGSCI commands `ADD` or `ALTER EXTRACT`. When `FILETYPE` is specified, Extract uses a series of application files as the source of database changes rather than TMF audit trails or log trails. `ALTINPUT` sets up processing rules for the source files.

Syntax

```
ALTINPUT [RANGE (x OF y)]
[, SHARED | EXCLUSIVE | PROTECTED]
[, OPENTIMEOUT minutes]
[, TEMPLATE template]
[, USENEXTMODIFIED]
```

```
[, NOWAITNEXTMODIFIED]
[, FASTREADS]
[, WAITNEXTRBA num_bytes]
[, JTSOFFSET byte_offset]
[, TANDEMTSOFFSET byte_offset]
[, ONEFILE]
```

RANGE (x OF y)
Use RANGE to process multiple files within a sequence simultaneously. For information on specifying the RANGE option see "Specifying the ALTINPUT RANGE Option"

SHARED | EXCLUSIVE | PROTECTED
Specifies the access mode of the OPEN for the input file. Defaults to SHARED, in read-only mode.

OPENTIMEOUT *minutes*
Determines when it is safe to assume that there are no more updates to the input file.

- If OPENTIMEOUT is not specified, Extract assumes there are no more updates five seconds after the last one is received.
- OPENTIMEOUT *minutes* instructs Extract to wait for the specified number of minutes before assuming there are no more updates.

An application can have an open file that is not being updated, such as when a report program has opened a file in other than read-only mode.

TEMPLATE *template*
Specifies a file name template when FILETYPE is ENTRY that can be a wildcard.
The following example specifies files on any volume starting with $DATA, on the MYDAT subvolume, beginning with FL and ending in exactly three more characters.

```
TEMPLATE $DATA*.MYDAT.FL???
```

In GGSCI, the ADD EXTRACT or ADD REPLICAT command could specify that the first file to process would be $DATA5.MYDAT.FLABC.

NOWAITNEXTMODIFIED
By default, Extract waits until the next file in the sequence is modified before processing it. Use NOWAITNEXTMODIFIED to move to the next file regardless of when it was modified.

FASTREADS
Causes Extract to perform bulk reads of the source file set, boosting read performance.

WAITNEXTRA *num_bytes*
Causes Extract to wait until the next file in the sequence has accumulated a specified number of bytes.

JTSOFFSET *byte_offset*
Specifies the byte offset of a 64-bit Julian timestamp field within each record when specifying FILETYPE ENTRY. Various Oracle GoldenGate processes use the Julian timestamp to help determine replication lag. Use a timestamp field that reflects when the record was inserted into the database.

TANDEMTSOFFSET *byte_offset*
Specifies the byte offset of a 48-bit timestamp field within each record when specifying `FILETYPE ENTRY`. Various Oracle GoldenGate processes use the Julian timestamp to help determine replication lag. Use a timestamp field that reflects when the record was inserted into the database.

ONEFILE
Specify `ONEFILE` if `FILETYPE ENTRY` is used and the file set to process consists of a single file.

Specifying the ALTINPUT RANGE Option

`ALTINPUT RANGE` allows processing of multiple ACI BASE24 files within a sequence simultaneously. Since two `TLF` or `PTLF` files can be active at the same time, using `ALTINPUT RANGE` enables one Extract to process even Julian date files and the other Extract to process odd Julian date files. Even and odd Julian dates can be determined by computing a Julian date from the last six digits of each file name.

The "even" Extract process retrieves data from files where the remainder of the Julian date divided by 2 is zero (range 1 of 2). The "odd" Extract retrieves data from files where the remainder of the Julian date divided by 2 is 1 (range 2 of 2).

For example, an "odd" Extract instance processes files named `PO990101`, `PO990103`, `PO990105` (Julian dates `2451181`, `2451183`, and `2451185`, respectively). An "even" Extract instance processes files `PO990102`, `PO990104`, `PO990106` (Julian dates `2451180`, `2451182`, and `2451184`, respectively). This enables extraction for files `PO990101` and `PO990102` at the same time.

> **Note:**
>
> The `RANGE` option of `ALTINPUT` should not be confused with the `RANGE` argument of `FILE` and `MAP` or the function `@RANGE` that can be used within a `FILTER`. The application of each one is different.

ASSUMETARGETDEFS

Valid for

Replicat

Description

Use `ASSUMETARGETDEFS` when the source and target files have the same record or columns structure (for example, in hot site replication). This parameter is useful when source and target data definitions match and a definitions file is not available for the source database.

Use `SOURCEDEFS` to use the source data definitions.

Syntax

```
ASSUMETARGETDEFS
```

AUDITREPS | NOAUDITREPS

Valid for

Replicat

Description

Use `AUDITREPS` to determine if Replicat transactions are framed within TMF transactions. It is highly recommended that you use `AUDITREPS` to ensure the integrity of the target database.

Use `NOAUDITREPS` when the target files are not protected by TMF. If any target tables or files are audited, `AUDITREPS` is required.

Default

`AUDITREPS`

Syntax

```
AUDITREPS | NOAUDITREPS
```

AUDITRETRYDELAY

Valid for

Extract

Description

Use `AUDITRETRYDELAY` to control the amount of time Extract waits at the end of TMF audit trails before attempting to read more data. Setting `AUDITRETRYDELAY` to a higher value can save system resources, but can also result in longer lag time when replicating data.

Default

`AUDITRETRYDELAY` 1

Syntax

`AUDITRETRYDELAY` *`seconds`* | `AUDITRETRYDELAYCSECS` *`centiseconds`*

`seconds`
The number of seconds to wait before reading more data.

`centiseconds`
The number of centiseconds to wait before reading more data.

AUDSERVCACHEBLOCKS

Valid for

Extract

Description

Use AUDSERVCACHEBLOCKS to determine the amount of cache space reserved for SQL table definitions. This parameter affects the amount of memory reserved by Audserv for SQL table definitions. Cache blocks are useful for processing SQL update statements quickly. The default is sufficient for most installations.

Default

300

Syntax

```
AUDSERVCACHEBLOCKS cache
```

cache
The amount of cache space. The maximum recommended value is 1000. At a minimum, allocate one cache block per frequently accessed table partition.

AUDSERVCPU

Valid for

Extract

Description

Use AUDSERVCPU to start the audit reading process on a different CPU from Extract. For example, AUDSERVCPU 1 starts Audserv for the master audit trail and any auxiliary trails in CPU 1. AUDSERVCPUS 3, 5, 6 starts Audserv for the master trail in CPU 3, for AUX01 in CPU 5, and for AUX02 in CPU 6.

Using AUDSERVCPU results in the constant gathering of audit while Extract performs its own processing. This technique can reduce batch run times by up to 20%.

When TMF uses auxiliary TMF audit trails, AUDSERVCPU can specify different processors for each Audserv process. If you specify fewer CPU than the number of master and auxiliary TMF audit trails, the last CPU specified is used as the default.

When you specify IGNOREAUXTRAILS, you should still specify AUDSERVCPU with the CPU for the ignored auxiliary trail as a placeholder for any subsequently included auxiliary trails.

Default

The last CPU specified

Syntax

```
AUDSERVCPU[S] cpu_num [, cpu_num...]
```

cpu_num
The CPU identifier.

AUDSERVPARAM

Valid for

Extract

Description

Use AUDSERVPARAM to pass parameters that are specific to the Audserv process.

> **Note:**
>
> AUDSERVPARAM GETPURGEDATAS and IGNOREPURGEDATAS options are deprecated. Extract will instruct Audserv to return purge data operations if any file or wildcard in its parameter file has GETFILEOPS or GETPURGEDATAS.

Default

```
ABENDONSECURITYCHECK, IGNORENONDATACHANGES, IGNOREALTFILES, SQLCATCLOSEDELAY 60
```

Syntax

```
AUDSERVPARAM
[ABENDONSECURITYCHECK | NOABENDONSECURITYCHECK]
[ARLIBERROR error_number, response]
[ARERRORREPORTINTERVAL seconds]
[GETNONDATACHANGES | IGNORENONDATACHANGES]
[GETALTFILES | IGNOREALTFILES]
[EXCLUDEFILECODES (file_code [, ...])
[SQLCATCLOSEDELAY seconds]
```

ABENDONSECURITYCHECK | NOABENDONSECURITYCHECK
With ABENDONSECURITYCHECK Audserv will log a message and then abend when the security check omits a file. NOABENDONSECURITYCHECK triggers Audserv to log a message, but not abend. The default is ABENDONSECURITYCHECK.

ARLIBERROR *error_number*, *response*
Specifies the action to take when the error number is triggered. Valid *response* are:

IGNORE
Continue processing and do not issue a message.

WARN
Issue a warning message and continue processing.

ABEND
Issue an error message and end processing.

ARERRORREPORTINTERVAL
Specifies the number of seconds to wait before the reissue of a warning message.

GETNONDATACHANGES | IGNORENONDATACHANGES
Filters records for SQL partition moves and splits. The default is IGNORENONDATACHANGES. NOGETNONDATACHANGES is a synonym for IGNORENONDATACHANGES.
When using NOGETNONDATACHANGES, you cannot extract any data changes produced by the RDF subsystem.

GETALTFILES | IGNOREALTFILES
By default Audserv excludes Enscribe alternate keys. This means, when a file is added to Audserv's "include" list, it will search for any alternate keys and automatically exclude them.
If you are using wildcards in your files to denote alternate keys, you may wish to override this feature. There are two ways to override the default programming. If you wish to include alternate keys for specific files, you may specify the following syntax in your parameter file:

```
FILE $VOL.SUBVOL.PRIMARY;
FILE $VOL.SUBVOL.ALTFILE0;
FILE $VOL.SUBVOL.ALTFILE1;
```

where ALTFILE0 and ALTFILE1 represent alternate keys in the PRIMARY file.
If you wish to replicate all alternate keys, you can use the GETALTFILES option in your Extract parameter file as follows:

```
AUDSERVPARAM GETALTFILES
```

The default setting for this option is IGNOREGETALTFILES.

EXCLUDEFILECODES (*file_code*)
The numeric file code of a type of file whose audit is to be excluded.
SQL/MX and SQL/MP catalog files (file codes 563, 564, 565, 572, and 585) are automatically excluded and do not need to be listed.

SQLCATCLOSEDELAY *seconds*
Sets the time delay after which the SQL/MP catalog tables that have not been accessed are closed. The value must be between 10 and 3600 seconds.
The default is to close the catalog tables when they have not been accessed for 60 seconds.

AUDSERVPREFIX

Valid for

Extract

Description

Use AUDSERVPREFIX to tell Extract to assign its Audserv process a sequence of names with the same prefix. The names are processed in the same order as MAT, AUX01, AUX02, and so on.

Syntax

```
AUDSERVPREFIX prefix
```

prefix
A 3-character prefix, as in $GAX.

Example

Specifying:

```
AUDSERVPREFIX $GAX
```

would assign:

```
MAT = $GAX00
AUX01 = $GAX01
AUX02 = $GAX02
```

If you use AUDSERVPROCESS, you can not use AUDSERVPREFIX, and vice versa.

AUDSERVPROCESS

Valid for

Extract

Description

Use AUDSERVPROCESS to control the names assigned to Audserv processes. The names are processed in the same order as MAT, AUX01, AUX02, and so on.

Syntax

```
AUDSERVPROCESS name
```

name
Assign a $5-character name

Example

Specifying:

```
AUDSERVPROCESS $GGMAT, $GGX01, $GGX02
```

results in the following assignments:

```
MAT - $GGMAT
AUX01 - $GGX01
AUX02 - $GGX02
```

If you use AUDSERVPROCESS, you can not use AUDSERVPREFIX, and vice versa.

AUDSERVPROGRAM

Valid for

Extract

Description

Use AUDSERVPROGRAM to run both Native mode and TNS mode Audserv programs. AUDSERVPROGRAM overrides the default program file name Audserv with the program name of your choice.

Syntax

```
AUDSERVPROGRAM program_name
```

Example

The following example starts Audserv using the program EXTRACTN and with the name AUDSERVN.

```
GGSCI> ADD EXT FINANCE, BEGIN NOW, PROCESS, PROGRAM EXTRACTN, DESCRIPTION "NATIVE
TMF EXTRACT"
GGSCI> ADD EXTTRAIL $DATA.GGSDAT.ET, EXTRACT FINANCE
EXTRACT FINANCE
AUDSERVPROGRAM $DATA.GGS.AUDSERVN
EXTTRAIL $DATA.GGSDAT.ET
FILE $PROD.ACCOUNT.*;
```

AUTORESTART

Valid for

Manager

Description

Use the AUTORESTART parameter to start one or more Extract and Replicat processes automatically after they fail. AUTORESTART provides fault tolerance when something temporary interferes with a process, such as intermittent network outages or programs that interrupt access to transaction logs.

You can use multiple AUTORESTART statements in the same parameter file.

Default

Do not auto-restart

Syntax

```
Syntax AUTORESTART process_type group_name
[, RETRIES max_retries]
[, WAITMINUTES wait_minutes]
[, RESETMINUTES reset_minutes]
```

process_type
Specify one of the following:

- EXTRACT or EXT
- REPLICAT or REP
- ER (Extract and Replicat)
- LOGGER
- SYNCFILE or SYNC
- COORD (Coordinator)

group name
A group name or wildcard specification for multiple groups. When wildcards are used, Oracle GoldenGate starts all groups of the specified *process_type* on the local system that satisfy the wildcard.

RETRIES ***max_retries***
The maximum number of times that Manager should try to restart a process before aborting retry efforts.
If RETRIES is not set, MAXABENDRESTARTS is used. If neither is set, the default number of tries is 2.

WAITMINUTES ***wait_minutes***
The amount of time to pause between discovering that a process has terminated abnormally and restarting the process. Use this option to delay restarting until a necessary resource becomes available or some other event occurs. The default delay is 1 minute.

RESETMINUTES ***reset_minutes***
The window of time a restarted process must run without abending for the retries count to be reset to the maximum. If the process abends within this time, the maximum retries value is decremented. When it reaches zero, no more restarts are attempted.
For example, RETRIES is set to 2 and RESETMINUTES is 15. If process A is restarted and runs without abending for 15 minutes RETRIES will be reset to 2. If instead, process A abends in less than 15 minutes, RETRIES becomes 1. If it is restarted and abends again within 15 minutes, no more retries will be attempted.
If RETSETMINUTES is not set for AUTORESTART, RESTARTINERVAL is used. If neither option is set, the default is 20 minutes.

Example

In the following example, Manager tries to start all Extract processes three times after failure within a one hour time period, and waits five minutes before each attempt.

```
AUTORESTART EXTRACT *, RETRIES 3, WAITMINUTES 5, RESETMINUTES 60
```

AUTOSTART

Valid for

Manager

Description

The AUTOSTART parameter specifies the processes to be automatically started by Manager. When Manager starts up it scans the list and attempts to start any listed process that is not already running. To activate changes to the list, stop and restart the Manager process.

Syntax

```
AUTOSTART [group_type] {group_name | process_name}
      [, ALLPROCESSES]
```

group_type
The type of group to be started. This is an optional entry, but if used must be one of the following.

- `EXTRACT or EXT`
- `REPLICAT or REP`
- `ER (Extract and Replicat)`
- `LOGGER`
- `SYNCFILE` or `SYNC`
- `COORD (Coordinator)`

group_name
The group name. Required entry for group types other than `Logger`. Wildcards can be used for part or all of the name.
Note: `TASK` groups that match the wildcard will not be started. To start a `TASK` group either specify a group without a wildcard or use the `ALLPROCESSES` option.

process_name
The `Logger` process name in the format `$xxnnn`. Required entry for the `LOGGER` *`group_type`*. Wildcards can be used for all or part of the name.

`ALLPROCESSES`
Specifies that `TASK` groups should be included in wildcarded groups to be started.

Examples

Example 1
The following will start all Replicat processes that begin with `R20`.

```
AUTOSTART REPLICAT R20*
```

Example 2
The following will start all Extracts.

```
AUTOSTART EXT *
```

Example 3
The following will start the logger named `$ABC01`.

```
AUTOSTART LOGGER $ABC01
```

Example 4
The following will start all groups that begin with `R20` except `TASK` groups.

```
AUTOSTART R20*
```

Example 5
The following will start all groups, including `TASK` groups, that begin with `R20`.

```
AUTOSTART R20*, ALLPROCESSES
```

BACKUPCPU

Valid for

Manager

Description

Use `BACKUPCPU` to specify a CPU that is running the backup Manager process.

Syntax

`BACKUPCPU` *`cpu_number`*

`cpu_number`
The identifier for the CPU that is running the backup Manager process.

BEGIN

Valid for

Extract, Replicat

Description

Use `BEGIN` to specify a date and time at which to start capturing data. Any record with a timestamp greater than or equal to the time specified by `BEGIN` that satisfies other criteria is extracted.

`SPECIALRUN` uses `BEGIN` to determine its start date and time.

Syntax

`BEGIN` *`date time`*`[:`*`seconds`*`][.`*`centiseconds`*`]`

`date`
The date in `yyyy-mm-dd` format.

`time`
The time in `hh:mm` format.

`seconds`
You can optionally specify seconds with the *`time`* option, as in `hh:mm:ss`.

`centiseconds`
You can optionally specify centiseconds seconds with the *`time`* and *`seconds`* options, as in `hh:mm:ss.cccccc`.

Examples

Example 1
`BEGIN 2010-08-12 08:00` specifies a timestamp for August 12, 2010 at 8:00 AM.

Example 2
`BEGIN 2010-08-12 08:00:30.000030` specifies a timestamp for August 12, 2010 at 8:00:30.000030 AM.

BULKIOLOAD | NOBULKIOLOAD

Valid for

Replicat

Description

Use BULKIOLOAD to enable bulk I/O, in 28K blocks, whenever writing unstructured data to structured or unstructured files. Typically, this occurs when replicating FUP LOAD or FUP DUP operations, and allows Replicat to process those types of operations many times faster than with conventional I/O.

BULKIOLOAD applies to all subsequent MAP entries. Use NOBULKIOLOAD to turn off BULKIOLOAD for subsequent MAP statements.

Default

NOBULKIOLOAD

Syntax

```
BULKIOLOAD | NOBULKIOLOAD
```

CHECKINTERVAL

Valid for

Syncfile

Description

Use CHECKINTERVAL to change the interval between the time a scheduled event occurs and the time that Syncfile duplicates an associated file. Once every minute, by default, Syncfile determines which of the scheduled events have occurred. When the event takes place, Syncfile duplicates the associated files.

Default

1 minute

Syntax

```
CHECKINTERVAL num_units unit;
```

num_units
The number of units of time.

unit
The time unit type. Specify one of: SECONDS, MINUTES or HOURS.

Example

```
CHECKINTERVAL 10 SECONDS;
```

CHECKMINUTES

Valid for

Manager

Description

Use CHECKMINUTES to determine how often Manager performs maintenance activities. If audit trails roll over frequently and the trails are actively managed, decreasing the frequency of maintenance activities can significantly affect performance.

Default

10

Syntax

```
CHECKMINUTES minutes
```

minutes
The frequency, in minutes, for performing maintenance.

Example

The following example specifies maintenance activities are performed every 20 minutes.

```
CHECKMINUTES 20
```

CHECKPARAMS

Valid for

Extract, Replicat

Description

Use CHECKPARAMS to verify parameter file contents before processing data. The program performs parameter checking, then quits before processing data.

Default

No parameter check

Syntax

```
CHECKPARAMS
```

COORDINATOR

Valid for

Coordinator, Replicat

COORDINATOR for Coordinator

Use COORDINATOR in the Coordinator parameter file to identify the name of the group.

Syntax

```
COORDINATOR group_name
```

group_name
The name of the group.

Example

```
COORDINATOR ACCTCO
```

COORDINATOR for Replicat

Use `COORDINATOR` in the Replicat parameter file to identify the name of the process that coordinates transactions distributed across multiple nodes in the network.

Syntax

```
COORDINATOR process_name
[MAXRETRIES number]
[DELAYSECS seconds | DELAYCSECS centiseconds]
```

process_name
The process name of the Coordinator group with which the Replicat will communicate to track distributed network transactions. The format for the default name is `\node.$GGCnn` with the node designator required.

MAXRETRIES *number*
The number of times Replicat will try to start the process before allowing the process to end abnormally. The default is 5.

DELAYSECS *seconds* | DELAYCSECS *centiseconds*
Sets the number of seconds or centiseconds that Replicat waits between tries. The default is 20 seconds.

Example

The following Replicat parameter specifies Coordinator group `$GGC00` on the `NY` system.

```
REPLICAT REPNET
COORDINATOR \NY.$GGC00
```

DOWNINFO

Valid for

Manager

Description

Use `DOWNINFO` to classify a process that has terminated abnormally as informational in the report generated by `DOWNREPORT`. When events are sent to the event log, they are reported as informational if either a process terminates normally or the `DOWNINFO` parameter is specified.

Manager sends both normal and abnormal stops to the event log. To exclude processes that have stopped normally from being sent, use the `IGNORESTOPPED` option.

See the `DOWNREPORT` parameter "DOWNREPORT" for information on setting the frequency for sending information to the event log.

Default

Manager reports a process that terminates abnormally as informational, but does not report a normally terminated process as informational.

Syntax

```
DOWNINFO [IGNORESTOPPED]
```

IGNORESTOPPED
Specifies that Manager should only report processes that abended, and should not report processes terminated using the normal `STOP` command.

NOFILEAGING

Valid for

Extract, Replicat

Description

Deactivates the default of aging files from memory at two day intervals of no usage. See FILEAGEDAYS

Default

```
FILEAGEDAYS 2
```

Syntax

```
NOFILEAGING
```

REVERSEWINDOWSECS | REVERSEWINDOWCSECS

Valid for

Replicat

Description

In both TMF and non-TMF applications, an anomaly can occur in which an insert and subsequent update to the same record appear in reverse order in the log trail or extract trail. When processing such reversed data, Replicat processes the update first and this results in a "record not found" error. `TMFEXCEPTIONS` can be used to hold the update in anticipation of the insert. With TMF audit trails, Replicat will hold such transactions until the end of the TMF transaction. With non-TMF audit trails, however, there is no TMF transaction, so the number of seconds or centiseconds needs to be set using one of the `REVERSEWINDOW` parameters.

`REVERSEWINDOW` enables Replicat to hold the transactions for a specified number of seconds or centiseconds (the reverse window) without attempting to process them. If a subsequent insert is found within the reverse window, the insert is applied to the target file followed by the update and no error is recorded. If the reverse window expires without finding a matching insert, the process abends or discards the transaction depending on the selected option.

REVERSEWINDOWSECS or REVERSEWINDOWCSECS alone will not trigger Replicat to hold exception transactions. For this to work you must also:

- Set TMFEXCEPTIONS on for that file.
- Use NOAUDITREPS to ensure a non-audited target.

If both REVERSEWINDOW and TMFEXCEPTIONS are specified, but NOAUDITREPS is not, a message is generated that REVERSEWINDOWSECS is ignored. TMFEXCEPTIONS will still be applied.

> **Note:**
>
> REVERSEWINDOWSECS should only be used when the source is a non-TMF audited database. It should not be used when the source is a TMF audit trail.

Default

ABENDEXPIREDWINDOW

Syntax

```
{REVERSEWINDOWSECS seconds | REVERSEWINDOWCSECS cseconds}
[ABENDEXPIREDWINDOW | DISCARDEXPIREDWINDOW]
```

REVERSEWINDOWSECS *seconds* | REVERSEWINDOWCSECS *cseconds*
REVERSEWINDOWSECS *seconds* specifies the wait time in seconds. The range is 1 to 30 seconds and the recommended setting is 1.
REVERSEWINDOWCSECS *cseconds* specifies it in centiseconds. The range is 10 to 100 centiseconds and the recommended setting is 10.

ABENDEXPIREDWINDOW
Causes Replicat to abend at the expiration of the wait time.

> **Note:**
>
> Using this option will cause Replicat to checkpoint to the current position in the trail.

DISCARDEXPIREDWINDOW
Causes Replicat to discard the expired record and continue. This may be the preferred option if it is important to minimize Replicat downtime and file inconsistencies can be corrected at a later time.

TABLESINCATALOG

Valid for

Extract

Description

The TablesInCatalog parameter retrieves the names of user tables that are registered in the specified catalog and builds file statements for them.

Syntax

```
TablesInCatalog <catalog name> [, VERBOSE ]
```

<catalog name>
<catalog name> is an SQL/MP catalog location as <$vol>.<subvol>

VERBOSE
VERBOSE lists the file statements names as they are created.

TCPIPSWITCHERRS

Valid for

Extract

Description

Use TCPIPSWITCHERRS to switch to alternate controllers and destination IP addresses when an error occurs. You can specify the number of retries to attempt before switching. The default is 5.

If you specify an alternate TCPIPPROCESSNAME and RMTHOSTALT, both the process and destination IP address are switched when the retry limit is reached.

Default

5

Syntax

```
TCPIPSWITCHERRS number_retries
```

number_retries
The number of retries.

Refer to "RMTHOSTALT" for an example using TCPIPSWTICHERRS.

TRACEALLOPENS

Valid for

Logger

Description

Use TRACEALLOPENS to record all open and close attempts on any file. This parameter instructs GGSLIB to send all open and close activity to Logger, including activity on files that are not on the extraction list.

Use the Logdump utility to examine these records and correlate them to Logger activity.

Default

Do not trace.

Syntax

```
TRACEALLOPENS
```

USEOBJECTDEFS

Valid for

```
GLOBALS
```

Description

Enables or Disables metadata from the entire install. Not recommend without consulting Oracle support.

Default

```
YES
```

Syntax

```
USEOBJECTDEFS YES | NO
```

VERSIONERR

Valid for

Extract

Description

Use `VERSIONERR` to specify error handling when database definitions are out-of-sync with audit trail records. Extract interprets data from the audit trails according to database definitions current at run-time. When a database definition changes during the course of an Extract run, data can be missed or misinterpreted.

For example, suppose a database column `NEWCOL` is added to table `TAB1` while Extract is running. Any rows added to `TAB1` after `NEWCOL` is added will contain a value for `NEWCOL`. However, since Extract is working with the old definition (before `NEWCOL`), `NEWCOL` values will be missed.

There are three situations to consider with respect to changing database definitions:

- The record being processed is an old version of the record. For example, the record is missing one or more columns that have since been added to the table. By default Extract considers this a normal condition and processes the record (`VERSIONERR OLD CONTINUE`).
- The record being processed is a new version of the record. More columns are present in the record than Extract expects. The danger here is that data can be

missed. By default, Extract terminates abnormally in this situation (`VERSIONERR NEW ABEND`) and can be restarted to pick up the new table definition and any new columns as well. However, if the target application has no need to use the new column, this may be undesirable.

- The record being processed is simply out of synchronization with the table definition according to Extract. The default response is to terminate abend (`VERSIONERR OUTOFSYNC ABEND`).

`VERSIONERR` lets you override these defaults by specifying custom responses to old, new and out-of-sync records.

Syntax

```
VERSIONERR type, {CONTINUE | WARN | DISCARD | ABEND}
```

type
The description of the record being processed. Valid values:

```
OLD
NEW
OUTOFSYNC
```

CONTINUE
Process the record as normal.

WARN
Process the record, but issue a warning to the error file.

DISCARD
Output the record to a discard file, but process more records.

ABEND
Terminate Extract abnormally. This option is appropriate when a new definition should be retrieved.

Example

```
VERSIONERR NEW, WARN
```

WARNRATE

Valid for

Replicat

Description

Use `WARNRATE` to set the rate at which SQL errors encountered are reported. For example, `WARNRATE 1000` issues a warning for every 1000 errors for SQL errors encountered for a particular target table. If many SQL errors are expected, `WARNRATE` helps minimize the size of the report file and error log.

Default

100

Syntax

```
WARNRATE num_errors
```

num_errors
The error report rate.

AUDITING

Valid for

GLOBALS

Description

Use AUDITING to automatically set other system parameters to TMF, non-TMF, or both. For example, setting AUDITING to NONTMF automatically sets the Replicat NOAUDITREPS parameter.

Default

ALL

Syntax

```
AUDITING {TMF | NONTMF | ALL}
```

CHECKPOINTSECS

Valid for

Extract and Replicat

Description

Use the CHECKPOINTSECS parameter to control how often Extract and Replicat make their routine checkpoints.

- Decreasing the value causes more frequent checkpoints. This reduces the amount of data that must be reprocessed if the process fails, but it could cause performance degradation because data is written to disk more frequently.
- Increasing the value causes less frequent checkpoints. This might improve performance, but it increases the amount of data that must be reprocessed if the process fails. When using less frequent Extract checkpoints, make certain that the transaction logs remain available in case the data has to be reprocessed.

> **Note:**
>
> In addition to its routine checkpoints, Replicat also makes a checkpoint when it commits a transaction.

This parameter is only valid when using log-based extraction. Avoid changing CHECKPOINTSECS unless directed to do so by Oracle GoldenGate Technical Support.

Default

10

Syntax

```
CHECKPOINTSECS seconds
```

seconds
The number of seconds to wait before issuing a checkpoint.

Example

```
CHECKPOINTSECS 20
```

CHECKUNIQUEKEY | NOCHECKUIQUEKEY

Valid for

Replicat

Description

Use `CHECKUNIQUEKEY` specify that the target file should be checked to ensure that the key does not already exist before inserting a record.

If a unique alternate key condition is violated while attempting to insert a record to an entry-sequenced Enscribe file, an "empty record" results in the target file.

Use `NOCHECKUNIQUEKEY` to reset `CHECKUNIQUEKEY` behavior for subsequent entries. `CHECKUNIQUEKEY` parameter affects only files with unique alternate keys. `CHECKUNIQUEKEY` applies only to `MAP` statements that follow it in the parameter file.

Default

```
NOCHECKUNIQUEKEY
```

Syntax

```
CHECKUNIQUEKEY | NOCHECKUNIQUEKEY
```

CLEANUPSAVECOUNT

Valid for

Manager

Description

Use `CLEANUPSAVECOUNT` to tell Manager how many old run history records to save for Extract and Replicat groups. Every evening after midnight, Manager cleans up old history records.

Default

10

Syntax

```
CLEANUPSAVECOUNT max_history_count
```

max_history_count
The number of history records to save. You can set the number of history records to a number between 5 and 50.

COBOLUSEREXIT

Valid for

Extract, Replicat

Description

Use `COBOLUSEREXIT` to call a custom COBOL routine at different points during processing.

If `COBOLUSEREXIT` is specified in the parameter file, but a user exit is not bound to the Extract or Replicat object, the process will abend.

Syntax

```
COBOLUSEREXIT
```

COLMATCH

Valid for

Extract, Replicat

Description

Use `COLMATCH` to map columns when source and target tables are different. The `COLMATCH` parameter enables mapping between databases with similarly structured tables but different names. `COLMATCH` specifies rules for default column mapping that apply to all columns that match the specified name.

`COLMATCH` is required when the source and target columns are different.You can also use the `COLMAP` option of the Replicat `MAP` parameter.

Syntax

```
COLMATCH
{NAMES target_column = source_column |
PREFIX prefix | SUFFIX suffix | RESET}
```

NAMES *target_column* = *source_column*
Matches a target column to a source column.

- *target_column* is the name of the target column.
- = is the assignment operator.

- *source_column* is the name of the source column.

PREFIX *prefix*
Specifies a prefix to ignore.

SUFFIX *suffix*
Specifies a suffix to ignore.

RESET
Turns off any COLMATCH rules previously specified.

Global rules and table names

It may be that a source and target database are identical except for slightly different names, as shown in the following table.

Source Database	Target Database
ACCT Table	ACCOUNT Table
CUST_CODE	CUSTOMER_CODE
CUST_NAME	CUSTOMER_NAME
CUST_ADDR	CUSTOMER_ADDRESS
PHONE	PHONE
ORD Table	ORDER Table
CUST_CODE	CUSTOMER_CODE
CUST_NAME	CUSTOMER_NAME
ORDER_ID	ORDER_ID
ORDER_AMT	ORDER_AMT

To map the source database columns to the target, you could specify each individual column mapping, but an easier method is to specify global rules using COLMATCH as follows:

```
COLMATCH NAMES CUSTOMER_CODE = CUST_CODE
COLMATCH NAMES CUSTOMER_NAME = CUST_NAME
COLMATCH NAMES CUSTOMER_ADDRESS = CUST_ADDR;
```

Specifying matches this way enables all columns in ACCT to be mapped to ACCOUNT, and all columns in ORD to map to ORDER. When performing default mapping, Extract checks for any matches according to global rules (this enables mapping of CUST_CODE, CUST_NAME and CUST_ADDR). In addition, exact column name matches are checked as always (enabling mapping of ORDER_ID, ORDER_AMT and PHONE).

Global rules and suffixes

Another frequently encountered situation is:

Source table	Target table
CUST_CODE	CUST_CODE_K
CUST_NAME	CUST_NAME_K
ORDER_ID	ORDER_ID
ORDER_AMT	ORDER_AMT

In this case, a global rule can specify that the `_K` suffix appended to columns in the target table be ignored, as in: `COLMATCH SUFFIX _K`.

This also resolves the opposite situation:

Source table	Target table
CUST_CODE_K	CUST_CODE
CUST_NAME_K	CUST_NAME
ORDER_ID	ORDER_ID
ORDER_AMT	ORDER_AMT

The same principle can be applied to column prefixes: `COLMATCH PREFIX P_`

Source table	Target table
P_CUST_CODE	CUST_CODE
P_CUST_NAME	CUST_NAME
ORDER_ID	ORDER_ID
ORDER_AMT	ORDER_AMT

Global rules and map entries

Global rules can be turned off for subsequent map entries with `COLMATCH RESET`.

COMMENT

Valid for

All

Description

Use `COMMENT` to insert comments within a parameter file. Anything on the same line after `COMMENT` is ignored during processing. Two hyphens (--) also denote a comment.

A comment can be entered anywhere within the parameter file. Comments continuing to the next line must be preceded by another double hyphen or `COMMENT` keyword.

If any columns in the tables being synchronized contain the word "comment," there may be conflicts with the `COMMENT` parameter, so double hyphens are the recommended option.

Syntax

```
COMMENT comment_text} | {-- comment_text}
```

Example

Both of the following are valid comments. The second uses the recommended syntax.

```
COMMENT Oracle GoldenGate NSK SQL Extract parameter file
-- Oracle GoldenGate NSK SQL Extract parameter file
```

COMPRESSDELETES | NOCOMPRESSDELETES

Valid for

Extract

Description

Use `COMPRESSDELETES` and `NOCOMPRESSDELETES` to control the way columns are written to the trail record for delete operations.

`COMPRESSDELETES` will extract only the primary key fields or columns for deleted operations. `NOCOMPRESSDELETES`, the default, sends all columns to the trail. By sending only the primary key, Oracle GoldenGate has all of the data required to delete the target record, while restricting the amount of data that must be processed. This creates a net performance gain.

`COMPRESSDELETES` and `NOCOMPRESSDELETES` can be used globally for all `TABLE` statements in the parameter file, or they can be used as on-off switches for individual `TABLE` statements.

Default

`NOCOMPRESSDELETES`

Syntax

```
COMPRESSDELETES
```

CONTROLTABLELOCKOFF

Valid for

Replicat

Description

`CONTROLTABLELOCKOFF` can be entered as a global level toggle before and after map statements as well as an option of a map.

Default

`NOCONTROLTABLELOCKOFF`

Syntax

```
NOCONTROLTABLELOCKOFF|CONTROLTABLELOCKOFF

replicat repxyz

CONTROLTABLELOCKOFF
map $data02.tssout.tableA, target $data03.tssout.tableA;
map $data02.tssout.tableB, target $data03.tssout.tableB;

or
```

```
replicat repxyz

map $data02.tssout.tableA, target $data03.tssout.tableA, controltablelockoff;

NOCONTROLTABLELOCKOFF
map $data02.tssout.tableB, target $data03.tssout.tableB;

CONTROLTABLELOCKOFF
map $data02.tssout.tableC, target $data03.tssout.tableC;
```

This parameter is to be used in the case where Replicat has a table lock granted preventing other applications from updating it and receiving error 73.

> **Note:**
>
> Depending on the number of records updated in a single transaction, the use of `MAXTRANSOPS` might be also required to avoid receiving error -8300, 35 (Open lock unit limit has been reached).

CONVERTALLFLOATSTOIEEE | NOCONVERTALLFLOATSTOIEEE

Valid for

Extract, Replicat

Description

Use `CONVERTALLFLOATSTOIEEE` to convert the Tandem (TDM) float data type numbers used by Oracle GoldenGate for HP NonStop to the Institute of Electrical and Electronics Engineers (IEEE) format used by Oracle GoldenGate for Windows and UNIX. `CONVERTALLFLOATSTOIEEE` converts all the float data type numbers in a file or table if the following conditions are met:

- There is no column mapping for the table or file
- Any SQL/MP table is not a view
- Definitions are provided for the Enscribe files

`CONVERTALLFLOATSTOIEEE` converts 64 bit float data types (SQL/MP type `float` or Enscribe type `double`.) It does not convert 32 bit float data types (SQL/MP `Real` or Enscribe type `float`) because these are converted by Oracle GoldenGate for Windows and UNIX.

Use `CONVERTALLFLOATSTOIEEE` and `NOCONVERTALLFLOATSTOIEEE` as toggles in the parameter file to turn the conversion on for some of the files or tables and off for others. A `CONVERTALLFLOATSTOIEEE` parameter will be in affect until a `NOCONVERTALLFLOATSTOIEE` parameter is encountered and vice versa.

> **Note:**
>
> To use CONVERTALLFLOATSTOIEEE, the Extract and Replicat must be native objects; not TNS.

Default

NOCONVERTALLFLOATSTOIEE

Syntax

CONVERTALLFLOATSTOIEE

Example

```
EXTRACT EXTORD
RMTHOST host01, MGRPORT 12345
CONVERTALLFLOATSTOIEEE
TABLE $DATA01.SALES.ORDERS;
NOCONVERTALLFLOATSTOIEEE
TABLE $DATA03.SALES.CUSTOMER;
```

CONVERTALLFLOATSTOTDM | NOCONVERTALLFLOATSTOTDM

Valid for

Extract, Replicat

Description

Use CONVERTALLFLOATSTOTDM to convert the Institute of Electrical and Electronics Engineers (IEEE) format used for float data type numbers from Oracle GoldenGate Windows and UNIX to the Tandem (TDM) format used by Oracle GoldenGate for HP NonStop. This will convert all the float data type numbers in a file or table if the following conditions are met:

- There is no column mapping for the file or table
- Any SQL/MP table is not a view
- Definitions are provided for the Enscribe files

CONVERTALLFLOATSTOTDM converts both 32 and 64 bit float data types because these are not converted by Oracle GoldenGate for Windows and UNIX.

Use CONVERTALLFLOATSTOTDM and NOCONVERTALLFLOATSTOTDM as toggles around the tables in the parameter file to turn the float data type conversion on for some of the tables and off for others. The NOCONVERTALLFLOATSTOTDM parameter will be in affect until the CONVERTALLFLOATSTOTDM parameter is encountered and vice versa.

> **Note:**
>
> To use `CONVERTALLFLOATSTOTDM`, the Extract and Replicat must be native objects; not TNS.

Default

```
NOCONVERTALLFLOATSTOTDM
```

Syntax

```
CONVERTALLFLOATSTOTDM
```

Example

```
REPLICAT REPORD
CONVERTALLFLOATSTOTDM
TABLE $DATA03.SALES.ORDERS,
  TARGET $DATA03.SALES.ORDERS;
NOCONVERTALLFLOATSTOTDM
TABLE $DATA03.SALES.ORDERS,
  TARGET $DATA03.SALES.CUSTOMER;
```

CPU

Valid for

Logger

Description

Use `CPU` to specify primary and backup CPUs for the current Logger process. For example: `CPU 9, 3` directs Logger to switch from the primary (`CPU 9`) to the backup (`CPU 3`) in the event of system problems.

The process on the primary CPU pings the process on the backup CPU every minute. If the backup CPU becomes unavailable, the primary process detects it and recreates it.

Both *`primary`* and *`backup`* are required

Syntax

```
CPU primary, backup
```

`primary`
The primary CPU identifier.

`backup`
The backup CPU identifier.

CUSEREXIT

Valid for

Extract, Replicat

Description

Use `CUSEREXIT` to call custom C routines at different points during processing. If your user exit is written in COBOL, see "COBOLUSEREXIT" for information on the `COBOLUSEREXIT` parameter.

If `CUSEREXIT` is specified in the parameter file, but a user exit is not bound to the Extract or Replicat object, the process will abend.

Syntax

```
CUSEREXIT
```

DEBUGONSTACKCHECK

Valid for

Logger

Description

Use `DEBUGONSTACKCHECK` to call `DEBUG` whenever an application's process stack is close to overflowing. Oracle GoldenGate recommends using this parameter only when instructed to do so by Oracle GoldenGate support.

Default

Omit `DEBUGONSTACKCHECK`

Syntax

```
DEBUGONSTACKCHECK
```

DECRYPTTRAIL

Valid for

Extract, Replicat

Description

Use `DECRYPTTRAIL` to decrypt Oracle GoldenGate trails that were encrypted by an upstream Extract process (for information on `ENCRYPTTRAIL` see "ENCRYPTTRAIL | NOENCRYPTTRAIL"). Specify `DECRYPTTRAIL` only when the `ENCRYPTTRAIL` parameter is specified in the upstream Extract process.

Syntax

```
Syntax
DECRYPTTRAIL
```

DICTIONARY

Valid for

Extract, Replicat

Description

Use `DICTIONARY` to establish an Enscribe DDL dictionary to use for evaluating `WHERE`, `FILTER`, and `COLMAP` clauses for `FILE` entries. For example, `DICTIONARY $DATA5.PRODDICT` specifies a physical subvolume.

Each `DICTIONARY` entry closes any previously open dictionary. This means only one is active at any given time while processing startup parameters. For this reason, you must specify your `DICTIONARY` parameter before the `FILE/MAP` entry that uses it.

Syntax

```
DICTIONARY dictionary_subvol
```

dictionary_subvol
A physical subvolume or an existing define name of class catalog.

DISCARDFILE

Valid for

Extract, Replicat

Description

Use `DISCARDFILE` to create a file containing records discarded by the process. Records can be discarded for a number of reasons, such as attempting to update a missing record. Each entry in the discard file provides the field names, field values and operation attempted, along with associated transaction information.

If `DISCARDFILE` is not present in the Replicat parameter file, Replicat will create a default discard file. See Monitoring Processing for details on the naming and characteristics of the default discard file.

Syntax

```
DISCARDFILE file_name
[, APPEND | PURGE] | ROLLOVER]
[, EXTENTS (primary, secondary, maximum) | MEGABYTES megabytes]
[, OWNER (group_number, user_number)]
[, SECURE "rwep"]
```

file_name
A physical file name or an existing define name of class map.

APPEND | PURGE | ROLLOVER

- APPEND - Appends records to an existing file. This is the default if no value is entered.
- PURGE - Purges an existing file and creates a new one
- ROLLOVER - Renames the discard files by appending a sequence number to the file name, according to these rules:

  If the file name is 7 characters or less, 1 digit is appended.

  If the file name is 8 characters, the file does not rollover and a warning is given that the nameis too long. This is true when the rollover is requested from GGSCi, as well as from the DISCARDFILE parameter.

EXTENTS (*primary*, *secondary*, *maximum*)
Sets the extent sizes and maximum extents for the file. The default setting is (4, 4, 100).

MEGABYTES *megabytes*
Sets the maximum size of the file in megabytes. The maximum size is 2 gigabytes.

OWNER (*group_number*, *user_number*)
Defines ownership of the discard file.

SECURE "*rwep*"
Secures the file using standard Guardian security for read, write, execute and purge operations.

Example

```
DISCARDFILE =DISCARD_FILE, OWNER 100,1, SECURE "NNNN", PURGE
```

DISCARDROLLOVER

Valid for

Extract, Replicat

Description

Use DISCARDROLLOVER to specify when a discard file is aged and a new one is created. Old files are renamed in the format of *group_name(n)*, where *group_name* is the name of the Extract or Replicat group and *(n)* is a number that gets incremented by one each time a new file is created, for
example: $DATA.GGSDISC.DISCARD0, $DATA.GGSDISC.DISCARD1, $DATA.GGSDISC.DISCARD2, and so forth.

Either the AT or ON option can be entered, or both options can be used together. The following rules apply to the possible variations in entries:

- Entering AT with a time but without a day creates a new discard file at the specified time every day.
- Entering AT without a time generates an error.
- Entering ON with a day but without a time creates a new discard file at midnight on the specified day.
- Entering ON without a day will generate an error.

- If no option is entered, the discard file will roll over at midnight every night.

To have more than one rollover, enter multiple `AT`, `ON`, or `AT ON` options. Up to 30 will be used.

A discard file will not roll over if:

- The discard file is empty
- The discard file was created by default rather than declared using the `DISCARDFILE` parameter

Default

Disabled. No rules specified.

Syntax

```
DISCARDROLLOVER {
AT hh:mm |
ON day_of_week |
AT hh_mm ON day_of_week
}
```

`AT` *`hh:mm`*
The time of day to age the file based on a 24-hour clock.
Valid values:

- *`hh`* is an hour of the day from 1 through 23.
- *`mm`* is minutes from 00 through 59.

`ON` *`day_of_week`*
The day of the week to age the file.
Valid values are `SUNDAY` through `SATURDAY`. They are not case-sensitive.

Examples

Below are examples of the use of `DISARDROLLOVER`.

Example 1
This example closes the existing discard file and creates a new one at 5:30 a.m. each day.

```
DISCARDROLLOVER AT 05:30
```

Example 2
This example rolls the discard file over every Friday at midnight.

```
DISCARDROLLOVER ON friday
```

Example 3
This example rolls the discard file over at 5:30 a.m. every Friday.

```
DISCARDROLLOVER AT 05:30 ON FRIDAY
```

Example 4
This example will roll over the discard file at 5:30 a.m. every Wednesday *and* at 5:30 am every Friday.

```
DISCARDROLLOVER AT 05:30 ON WEDNESDAY, AT 05:30 ON FRIDAY
```

DISKTHRESHOLD | NODISKTHRESHOLD

Valid for

Manager

Description

Use `DISKTHRESHOLD` to generate an event message when the percentage of all audit left on disk falls below the specified percentage. Use `NODISKTHRESHOLD` to eliminate reporting disk related thresholds.

Because audit is always being created, old audit eventually has to be recycled. If an Extract process needs audit that is about to be recycled, processing integrity is in danger.

To report audit thresholds, specify a percentage level for `DISKTHRESHOLD` in the Manager parameter file. When Extract program processing falls to the specified percentage, Manager sends a threshold message to EMS or to another specified location at regular intervals.

For example: You have ten audit trail files numbered 10-19, and you specify `DISKTHRESHOLD 25` to generate an event message when audit files to process reaches 25% of the entire audit trail. When Extract's checkpoint is positioned in audit file 10 or 11, Manager generates a threshold message, because less than 25% of audit available for writing new data before the current position is recycled.

Alternatively, you can use the `THRESHOLD` or `NOTHRESHOLD` parameters to report on disk or tape audit thresholds. For information on these parameters see "THRESHOLD | NOTHRESHOLD".

Default

```
DISKTHRESHOLD 20%
```

Syntax

```
DISKTHRESHOLD percent_left | NODISKTHRESHOLD
```

percent_left
The percentage level at which to generate an event message.

DISPLAYFILEREFRESHES

Valid for

Extract, Replicat

Description

Use `DISPLAYFILEREFRESHES` to display a message when attributes are refreshed for a file. `DISPLAYFILEREFRESHES` is a global parameter that is set once for the parameter file. It triggers generation of a refresh message when file attributes for each file included in the processing are refreshed.

Default

File refresh messages are suppressed.

Syntax

```
DISPLAYFILEREFRESHES
```

Example

When DISPLAYFILEREFRESHES is used, each file refresh displays a message similar to:

```
File Attributes Refreshed for
\NODE1.$VOL1.SUBVOL.FILE1 at 2013-05-13 09:13:36
```

DISPLAYTRAILSWITCH | NODISPLAYTRAILSWITCH

Valid for

Extract

Description

Use DISPLAYTRAILSWITCH or NODISPLAYTRAILSWITCH to print or suppress printing of messages to the report file when Extract switches trails.

Default

```
DISPLAYTRAILSWITCH
```

Syntax

```
DISPLAYTRAILSWITCH | NODISPLAYTRAILSWITCH
```

DOWNCRITICAL

Valid for

Manager

Description

Use DOWNCRITICAL to classify a process that has terminated normally as critical in the report generated by DOWNREPORT. When events are sent to the event log, they are reported as critical if either a process terminates abnormally or the DOWNCRITICAL parameter is specified.

Manager sends both normal and abnormal stops to the event log. To exclude processes that have stopped normally from being sent, use the IGNORESTOPPED option.

See the DOWNREPORT parameter "DOWNREPORT" for information on setting the frequency for sending information to the event log.

Default

Manager reports a process that terminates abnormally as critical, but does not report a normally terminated process as critical.

Syntax

```
DOWNCRITICAL [IGNORESTOPPED]
```

IGNORESTOPPED
Specifies that Manager should only report processes that abended, and should not report processes terminated using the normal STOP command.

DOWNREPORT

Valid for

Manager

Description

Use DOWNREPORTMINUTES or DOWNREPORTHOURS to report Extract and Replicat groups that are not running every *n* minutes or hours.

Events are generated any time Extract and Replicat processes are started, stopped, or interrupted. The Manager reports when a process is terminated.

Default

One hour

Syntax

```
DOWNREPORTMINUTES minutes | DOWNREPORTHOURS hours
```

minutes
The reporting interval, in minutes.

hours
The reporting interval, in hours.

DUP

Valid for

Syncfile

Description

Use DUP to specify the source and target file sets to duplicate and the name of the event to execute. Optionally, you can exclude individual files and/or types of files from duplication, duplicate based on whether records have changed, and execute FUP or TACL commands.

Follows the EVENT parameter. At least one DUP entry is required.

Syntax

```
DUP source_file_set, TARGET target_file_set
{, EVENT event_name}
[, NAME identifier]
```

```
[, EXCLUDE exclude_file],
[, INCLUDEFILECODE (include_codes)
[, EXCLUDEFILECODE (exclude_codes)],
[, CHANGED | ALWAYS]
[, FUPOPTIONS "fup_options"]
[, GETAUDITED | IGNOREAUDITED]
[, TACLCMD tacl_command];
```

source_file_set
Required. Identifies the source file set. You can use standard wildcards.

TARGET *target_file_set*
Required. Identifies the target file set. You can use standard wildcards.

EVENT *event_name*
Required. Specifies an event that has been defined by the EVENT parameter. You can specify multiple events as:

```
EVENT event_name, EVENT event_name,...
```

NAME *identifier*
Optional logical name to be assigned to the DUP. Enclose in quotes if the name contains spaces. If NAME is not specified, the logical name will default to ID_*num* where *num* is the ordinal number of the DUP item within the parameter file. The logical name is displayed when Syncfile starts processing a DUP.

EXCLUDE *exclude_file*
Specify EXCLUDE when you want to exclude certain files in the file set from duplication.

INCLUDEFILECODE (*include_codes*)
A comma delimited list to specify the file codes to be excluded from duplication. Wildcards are not accepted. The list must be enclosed in parenthesis, for example:

```
INCLUDEFILECODE (32767, 65000)
```

If an INCLUDEFILECODE list is specified, any file code that is not in the list will be excluded.

EXCLUDEFILECODE (*exclude_codes*)
A comma delimited list to specify the file codes to be included in the duplication. Wildcards are not accepted. The list must be enclosed in parenthesis, for example:

```
EXCLUDEFILECODE (0, 100, 700, 800)
```

If a file code is on the EXCLUDEFILECODE list it will be excluded, even if it is on the INCLUDEFILECODE list.

CHANGED | ALWAYS

- CHANGED is the default. Duplicates files only if the source file has changed more recently than the target file.
- ALWAYS duplicates source files whether or not source records have been modified since the corresponding target file records.

FUPOPTIONS "*fup_options*"
Enables options to be appended to the FUP DUP command. By default, Syncfile executes:

```
FUP DUP source, target, PURGE, SAVEALL
```

You can replace the PURGE and SAVEALL options with others. For example: specifying: FUPOPTIONS "SOURCEDATE, NEW"; results in the command:

```
FUP DUP source, target, SOURCEDATE, NEW.
```

GETAUDITED | IGNOREAUDITED

- GETAUDITED duplicates files that have the audit flag set to on.
- IGNOREAUDITED duplicates files that have the audit flag set to off. IGNOREAUDITED is the default.

TACLCMD *tacl_command*
See "Specifying TACL commands".

Specifying TACL commands

The TACLCMD *tacl_command* option executes a user-supplied TACL macro to perform the duplication. The macro can execute virtually any set of Guardian functions. For example, you can write a TACL macro to make a copy of the file to duplicate to a temporary location; edit the temporary file and change occurrences of the string "$DATA3" to "$D16"; age the previous file on the secondary system to a backup location; and FTP the temporary file to the backup.

When specifying TACLCMD, enclose the entire command to execute in quotes as shown below:

```
TACLCMD "RUN $DATA1.TACLMAC.DUPMAC source target"
```

As part of this command, you can specify the source and target file names as the *source* and *target* arguments. For example, the following command causes Syncfile to invoke TACL with $D16.TEST.CFG1 as the source and \BKUP.$DATA6.TEST.CFG1 as the target:

```
RUN $DATA1.TACLMAC.DUPMAC $D16.TEST.CFG1 \BKUP.$DATA6.TEST.CFG1
```

Enter <source> and <target> without substituting any file names to trigger Syncfile to take the source and target arguments from the DUP statement. For example, the following command will duplicate $DATA1.GGSPARM.* to $DATA5.GGSPARM.*.

```
DUP $DATA1.GGSPARM.*, TARGET $DATA5.*.*,
TACLCMD "RUN $DATA1.GGSPARM.TACL5 <SOURCE> <TARGET>",
ALWAYS, EVENT DAILY 1330;
```

DUPONLYAFTEREVENT

Valid for

Syncfile

Description

Use DUPONLYAFTEREVENT to change the time events execute.

As part of the EVENT definition, you can specify that events should execute at a certain time each day (for example, 01:00). By default, Syncfile executes an event if the current time is greater than time specified, and the event has not yet been performed

that day. Therefore, if Syncfile is started at 11:00, an event specified at 01:00 executes when Syncfile starts and then not again until 01:00 the following day.

In some cases you may want the event to execute later than the intended time. Specifying DUPONLYAFTEREVENT executes the event after the specified time is passed, ensuring that DUPs happen close to the intended time.

DUPONLYAFTEREVENT follows the EVENT parameter.

Syntax

```
DUPONLYAFTEREVENT;
```

DUPPROCESS

Valid for

Syncfile

Description

Use DUPPROCESS to specify the process name used by the TACL or FUP process used to duplicate files. This is generally for debugging purposes.

You must specify either the FUPOPTIONS or TACLCMD options for the DUP parameter.

Syntax

```
DUPPROCESS process_name;
```

process_name
The process name used to duplicate files.

DYNAMICPARTITIONS

Valid for

Extract

Description

Use DYNAMICPARTITIONS if your environment dynamically partitions tables; this ensures all data is captured.

By default, Extract scans tables on startup to determine the base table, primary partition, and any secondary partitions containing data, so it can begin its captures. However, if your system dynamically creates another partition, Extract does not know it exists unless the process stops, starts, and conducts another scan.

Using the DYNAMICPARTITIONS parameter allows Extract to recognize additional secondary partitions without having to stop and restart. This means that any data stored in the newly-created partition will be captured as part of Extract's regular processing.

For example, Extract is set up to look at $DATA2.SQL.ACCOUNT, and includes DYNAMICPARTITIONS in its parameter file. $DATA2.SQL.ACCOUNT is a table with a single secondary partition, $DATA4.SQL.ACCOUNT. When Extract starts, it sees

both $DATA2.SQL.ACCCOUNT and $DATA4.SQL.ACCOUNT, and records are captured from both. As processing continues, a new partition called $DATA6.SQL.ACCOUNT is created. Because the DYNAMICPARTITIONS parameter is specified, data stored in $DATA6.SQL.ACCOUNT is captured as well.

> **Note:**
>
> Do not use DYNAMICPARTITIONS if the same subvolume and file combination exists more than once in your system as primary partitions.

Default

Not activated

Syntax

```
DYNAMICPARTITIONS
```

DYNAMICPORTLIST

Valid for

Manager

Description

Use DYNAMICPORTLIST to specify the ports that Manager can dynamically allocate to Collector and Replicat processes and to GGSCI sessions. You can specify ports individually or a range of ports.

> **Note:**
>
> The DYNAMICPORTLIST is used by the Manager only for processes that are started dynamically, such as a Collector. Processes that are started by the user, such as a typical Extract, do not draw from this list.

When specifying individual ports, delimit each port with a comma. To specify a range of ports, use a dash (-) to separate the first and last port in the range. You can combine a range of ports and individual ports in the same statement.

Syntax

```
DYNAMICPORTLIST {port | port-port} [, ...]
```

port
A port (or ports) that can be dynamically allocated. Port entries are limited to 256.

Example

```
DYNAMICPORTLIST 7820 - 7830, 7833
```

DYNAMICPORTREASSIGNDELAY

Valid for

Manager

Description

Use `DYNAMICPORTREASSIGNDELAY` to specify the time to wait before a port can be reused.

Default

3

Syntax

`DYNAMICPORTREASSIGNDELAY` *`time`*

`time`
The number of seconds to delay before reusing a port.

EMBEDDEDMACROS | NOEMBEDDEDMACROS

Valid for

Extract, Replicat

Description

Use `EMBEDDEDMACROS` to control whether a macro can be expanded in a quoted string. Use `NOEMBEDDEDMACROS` to make text inside a quoted string invisible.

Default

`NOEMBEDDEDMACROS`

Syntax

`EMBEDDEDMACROS | NOEMBEDDEDMACROS`

EMSLOG

Valid for

`GLOBALS`

Description

Use `EMSLOG` to direct EMS messages to a Collector other than the default (`$0`).

The Extract, Replicat, and Manager programs check for the NonStop define name `=EMS_COLLECTOR`. As part of process initialization these programs take the value set for `EMSLOG` and use it to override any pre-existing value that was set for `=EMS_COLLECTOR`.

When Manager creates a process, it sets the NonStop define values for `=EMSCOLLECTOR` and `=EMS_COLLECTOR` to the value specified for `EMSLOG` (or the default of `$0`).

Default

$0

Syntax

```
EMSLOG {collector | NONE}
```

collector
Specify either the Collector name, or `NONE` when there is no Collector.

ENCRYPTTRAIL | NOENCRYPTTRAIL

Valid for

Extract

Description

Use `ENCRYPTTRAIL` to encrypt data records in subsequent Oracle GoldenGate trails until a `NOENCRYPTTRAIL` is encountered. All records going into an Oracle GoldenGate trail are encrypted both across any data links and within the trail. This applies to `EXTFILE`, `EXTTRAIL`, `RMTFILE` and `RMTTRAIL` entries. `ENCRYPTTRAIL` is not recommended for `RMTBATCH`.

Parameter files for downstream Extract or Replicat processes must specify a `DECRYPTTRAIL` to read the files.

Default

```
NOENCRYPTTRAIL
```

Syntax

```
ENCRYPTTRAIL | NOENCRYPTTRAIL
```

END

Valid for

Extract, Replicat

Description

Use `END` to specify the point at which the process stops processing in the TMF audit or Oracle GoldenGate trails. If `END` is omitted, processing continues until you stop it manually with GGSCI.

With `END`, processing terminates when an audit record is encountered with a timestamp equal to or greater than the time specified. You can specify the day, the time of day, including seconds and centiseconds as in: `END 2010-08-12 17:00:00`.

`END` is used to determine when `SPECIALRUN` processing will terminate.

Syntax

```
END {date [time] | RUNTIME}
```

date time
Causes Extract or Replicat to terminate when it encounters a record with a timestamp equal to, or greater than, the time specified.
Valid values:

- *date* is a date in the format of `yyyy-mm-dd`.
- *time* is the time in the format of `hh:mi[:ss[.cccccc]]` based on a 24-hour clock.

RUNTIME
Causes Extract or Replicat to terminate when it reaches process startup time. One advantage of using `RUNTIME` is that you do not have to alter the parameter file to change dates and times from run to run.

ENTRYSEQUPDATES

Valid for

Replicat

Description

Use `ENTRYSEQUPDATES` to enable entry-sequenced records to be replicated with exactly the same key as the record on the source system. Because standard Guardian functions do not permit control of entry-sequenced record keys, omitting `ENTRYSEQUPDATES` results in the following limitations:

- During an initial load (or similar situation), duplicate record conditions may not be detected; therefore, multiple instances of the same record can occur in the target file.
- When updates occur on the source database, there is no guarantee that the corresponding key on the target database is the same. Therefore, to guarantee updates are applied properly, a unique alternate key must be specified.

When `ENTRYSEQUPDATES` is specified, Replicat manipulates file contents directly through unstructured file access. This type of access imposes the following restrictions on the target file:

- The file is open for `PROTECTED` access (no other processes, including other Replicat processes, can update the file).
- The target file cannot be TMF audited.
- No more than one source file can be associated with the target file (to guarantee that the keys are unique).
- A trade-off exists between flushing file contents at any given time and performance.

`ENTRYSEQUPDATES` causes Replicat to manipulate file blocks directly. This ensures that each record is inserted into the target file in exactly the same place as in the source file (for other file types, this occurs automatically). To maximize performance, Replicat attempts to build these blocks in memory, keeping a private cache, and assumes

subsequent records will be inserted in close proximity. This maximizes performance by minimizing the amount of messages with the disk process.

You must specify ENTRYSEQUPDATES in the parameter file before any MAP statements for entry-sequenced files.

Syntax

```
ENTRYSEQUPDATES {EXACTKEY | NOEXACTKEY}
[, FLUSHALWAYS | NOFLUSHALWAYS]
[, HIDEGAPS | NOHIDEGAPS]
[, EXCLUSIVEOPEN | PROTECTEDOPEN]
```

EXACTKEY | NOEXACTKEY
Subsequent maps should always have exact key replication specified. Specify NOEXACTKEY to cancel exact key replication for subsequent entries.

FLUSHALWAYS | NOFLUSHALWAYS
Each time a block is updated, it is flushed to disk. This makes the record immediately visible to the application. The default is NOFLUSHALWAYS.

HIDEGAPS | NOHIDEGAPS

- HIDEGAPS — The default. If records for a particular block arrive out of order, hide records in the block from view until all prior records in the block arrive. This guarantees that records with a length of zero are never returned. Use HIDEGAPS when replicating non-TMF entry-sequenced files.
- NOHIDEGAPS — Make visible all records in a block, regardless of whether prior records have arrived. The trade-off is that missing records will appear to have a length of zero when reading the file.

EXCLUSIVEOPEN
Opens the target file exclusively. This is recommended by Oracle GoldenGate when opening structured files for unstructured write access, as is done when ENTRYSEQUPDATES is specified.

PROTECTEDOPEN
Allows users access to read-only files while Oracle GoldenGate is writing to the file.

Example

The following example causes exact replication of the key, optimized for the highest throughput.

```
ENTRYSEQUPDATES, EXACTKEY, NOFLUSHALWAYS
```

EOFDELAY | EOFDELAYCSECS

Valid for

Extract, Replicat

Description

Use EOFDELAY or EOFDELAYCSECS to specify the number of seconds or centiseconds to delay before looking for more data. Increase the time interval to increase the lag time

between updates on the source and target systems, especially when the source system is experiencing a small amount of activity.

This parameter only applies when Extract or Replicat is reading an Oracle GoldenGate trail.

Default

1 second

Syntax

```
EOFDELAY seconds | EOFDELAYCSECS centiseconds
```

seconds
The number of seconds to delay.

centiseconds
The number of centiseconds to delay.

ERREPLYTIMEOUT

Valid for

`GLOBALS`

Description

Use `ERREPLYTIMEOUT` to set the timeout, in seconds, when GGSCI communicates with Oracle GoldenGate components.

Default

30

Syntax

```
ERREPLYTIMEOUT seconds
```

seconds
Specify a timeout value in seconds.

ERROR59ROLLOVER

Valid for

Extract

Description

Use `ERROR59ROLLOVER` during direct file extraction to tell Extract to skip ahead to the next file in a sequence upon encountering a damaged block (Guardian Error 59).

Certain applications leave file blocks in a damaged state to signal that the end of data within the file has been reached. All data before the first damaged block is considered valid. Extract, when processing a sequence of files directly, will correctly process all

data up until the damaged block. By default, if Extract subsequently reads a damaged block, it will abend.

Syntax

```
ERROR59ROLLOVER
```

ETNEWFORMAT

Valid for

Extract

Description

Use `ETNEWFORMAT` to cause Extract to generate trails in formats that are compatible with Replicat version 7.0 or later.

Default

```
ETNEWFORMAT
```

Syntax

```
ETNEWFORMAT
```

EVENT

Valid for

Syncfile

Description

Use `EVENT` to define an event and its schedule. Associate the event with a file set through the `EVENT` keyword in `DUP` parameter argument. For example, you might define an `EVENT` named `DAILY` that executes a duplication at 03:00 daily:

```
EVENT DAILY,
EVERY DAY AT 03:00;
.
.
.
DUP $DATA1.GGSCFG.Z*, TARGET \BKUP.$DAT2.*.*,
ALWAYS
EVENT DAILY;
```

For this example, the `DUP` arguments includes `EVENT DAILY`. At 03:00, every day, data set `$DATA1.GGSCFG.Z*` is duplicated to data set `\BKUP.$DAT2.*.*`.

You can include multiple `EVENT` parameters. You can schedule for time of day, day of week, or date. You can also exclude specific dates or days of the week.

`EVENT` precedes all other parameters. At least one `EVENT` entry is required.

Syntax

```
EVENT event_name, every_options [, exclude_options];
```

event_name
Any name to identify the `EVENT`. The name is used by subsequent `DUP` entries to link file set duplication to an appropriate schedule.

every_options
Can be one of the following:

`EVERY DAY AT` ***`time`***
Enter the time as `0100`, `0200`...`1400`, `1500`, etc.

`EVERY` ***`date`*** **`AT`** ***`time`***
Specify the month and day of the month, as in: August 3. Month is spelled out. Enter the time as `01:00`, `02:00`...`14:00`, `15:00`, etc.

`EVERY` ***`time_interval`***
Enter an interval, such as `2 HOURS` or `10 MINUTES`.

exclude_options
Can be one or both of the following:

`EXCLUDE` ***`day_of_week`***
Spelled out, such as `SUNDAY` or `SATURDAY`.

`EXCLUDE` ***`date`***
Specify the month and day of the month, as in: `August 3`. Month is spelled out.

Example

For this example, two `EVENTS` are defined, then called by subsequent `DUP` parameters.

```
EVENT DAILY,
EVERY DAY AT 1:00;
EVENT FREQUENT,
EVERY 2 HOURS;

DUP $DATA1.GGSCFG.Z*, TARGET \BKUP.$DATA2.*.*,
ALWAYS,
EVENT DAILY;

DUP $DATA1.GGSPARM.*, TARGET \BKUP.$DATA2.*.*,
EVENT FREQUENT;
```

EXCLUDEFILE

Valid for

Logger

Description

Use `EXCLUDEFILE` to exclude specific files from extraction to the current log. You can implicitly and explicitly exclude files that have been included by the `FILE` parameter. For example, if the Logger parameter file includes `FILE` and `EXCLUDEFILE` entries similar to:

```
FILE $DATA4.*.*
EXCLUDEFILE $DATA4.DAT.TRANSFL
```

The EXCLUDEFILE parameter excludes the specified file set, even though the preceding FILE statement implicitly included it with a wildcard argument.

If used, EXCLUDEFILE must follow the FILE parameter. If a file is included by the FILE parameter without filters, then excluded by EXCLUDEFILE according to one or more filters, the file is excluded when the filter criteria are met.

You can exclude files by:

- Specifying file sets to exclude.
- Filtering file sets that are opened by a specified process or program.
- Filtering file sets that are associated with an opening program's user ID.

Syntax

```
EXCLUDEFILE file_set
[, PROCESS process_set]
[, PROGRAM program_set]
[, USER user_set]
```

file_set
The name of the file set to exclude.

PROCESS *process_set*
Excludes data when the opener is the process or set of processes specified (process set can be a single process or a wildcard, for example $APP*).

PROGRAM *program_set*
Excludes data when the opener is the program or set of programs specified (program set can be a single program or a wildcard: $DATA1.PROGS.*).

USER *user_set*
Excludes data when the creator access ID of the opener is the user specified (user set can be a single user or wildcard: FINANCE.JOE or SUPER.*).

Examples

Example 1

```
EXCLUDEFILE $DATA4.DAT.TRANSFL
```

Example 2

```
EXCLUDEFILE $D15.DAT.*
```

EXCLUDEGGSTRANSRECS | INCLUDEGGSTRANSRECS

Valid for

Extract

Description

Use INCLUDEGGSTRANSRECS to create network transaction tracking records. You need these records if you replicate distributed network transactions using Readers and a Coordinator. These processes depend on tracking records in the trail.

The default EXCLUDEGGSTRANSRECS suppresses the creation of network transaction tracking records.

Default

EXCLUDEGGSTRANSRECS

Syntax

INCLUDEGGSTRANSRECS

EXCLUDESUFFIXCHAR

Valid for

Manager

Description

Use EXCLUDESUFFIXCHAR to specify characters that are not to be used as the first character of the suffix when generating process names. This can be useful in avoiding conflicts with process naming conventions already in use.

The process name is made up of a two-character prefix, which can be set using ADD DEFINE for GGS_PREFIX (see "ADD DEFINE"), and a three character suffix. The suffix can have one or two alphanumeric characters and will end with a sequential number. The process name $GGL00, for example, uses the prefix GG. The first character of the suffix is L, indicating a Logger process, and the zeroes indicate it is the first process generated for $GGL.

Characters that are used to name Oracle GoldenGate processes are excluded by default. These include C (Coordinator), E (Extract), L (Logger), R (Replicat), and S (Server/Collector). The characters, MG, used in naming Manager processes, are also excluded by default.

Default

CELRS and MG

Syntax

EXCLUDESUFFIXCHAR *characters*

characters
The characters to be excluded. These should be entered as a string of characters without commas. The characters can optionally be enclosed in single or double quotation marks (e.g. "WTZ" or 'BP').

Examples

Example 1
The following examples all exclude the characters T and X from the first position of the suffix for generated process names.

```
EXCLUDESUFFIXCHAR TX
EXCLUDESUFFIXCHAR 'TX'
EXCLUDESUFFIXCHAR "TX"
```

Example 2
To reset to the defaults, enter an empty set of characters as shown below.

```
EXCLUDESUFFIXCHAR ""
```

EXPANDDDL

Valid for

Extract, Replicat

Description

Use EXPANDDDL to format Enscribe data. Enscribe DDL definitions frequently contain occurs, or array items. For example, a definition might contain:

```
05 GROUP1 OCCURS 3 TIMES.
   10 FIELD1 PIC X(5) OCCURS 20 TIMES.
```

To reference items within arrayed structures in a WHERE, FILTER, or COLMAP clause, you must identify the occurrence. The default syntax for doing so is: *field_name-occurrence*. For example, to retrieve the second occurrence of FIELD1 in the third group, the syntax would be GROUP1-3.FIELD1-2

The EXPANDDDL parameter changes this array notation. For example:

- EXPANDDDL USEBRACKETS specifies field as GROUP1[3].FIELD1[2]
- EXPANDDDL USETWOUNDERSCORES, ZEROFILL ARRAYWIDTH specifies field as GROUP1__3.FIELD1__02, when the maximum array width in GROUP1 is less than 10 (requiring at most one digit) and the maximum array width of FIELD1 is 20 (requiring two digits).

EXPANDDDL also determines how field occurrences are output when or FORMATSQL are specified.

Syntax

```
EXPANDDDL format
[, ZEROFILL width | ARRAYWIDTH]
[, INCLUDEREDEFS | OMITREDEFS]

[, RESOLVEDUPINDEX]
[, RESOLVEDUPGROUP]
[, EXPANDGROUPARRAYS]
```

format
Can be one of the following:

- USEDASH — Reference array items by -n, where n is the occurrence number.
- USEBRACKETS — Reference array items by [n], where n is the occurrence number.
- USEUNDERSCORE — Reference array items by _n, where n is the occurrence number.
- USETWOUNDERSCORES — Reference array items by __n, where n is the occurrence number.

ZEROFILL *width* | ARRAYWIDTH
Directs Extract to reference occurrences of each field adjusting for a maximum width.

INCLUDEREDEFS | OMITREDEFS
INCLUDEREDEFS includes redefined fields.
OMITREDEFS is the default. It excludes redefined fields, which has the following consequences:

- Data is only output to columns that do not redefine another field.
- When Extract specifies FORMATASCII or FORMATSQL, Extract does not output redefined fields.

RESOLVEDUPINDEX
Appends a numerical index to the end of each duplicate field or column. For example, END-DATE.YY would become YY-2 in the output since it is the second occurrence of YY in the definition.

RESOLVEDUPGROUP
Prefixes a duplicate field name with its group name and separator.

EXPANDGROUPARRAYS
Appends indexes to fields that do not necessarily occur multiple times, but which are part of groups that occur multiple times.

EXTFILE

Valid for

Extract, Replicat

Description

Use EXTFILE when Extract writes to an Oracle GoldenGate trail that contains a single file. EXTFILE defines the name, dimensions, and security of a file in the Oracle GoldenGate trail. The parameter file must include at least one EXTFILE or EXTTRAIL entry. EXTFILE must precede the names of files and tables containing data you want to extracted into the file.

All FILE and TABLE entries after the current entry but before the next EXTFILE, EXTTRAIL, RMTFILE, or RMTTRAIL parameter result in output to the current trail.

The trail must contain record headers or an error is returned at run-time.

Syntax

```
EXTFILE file_name
[, APPEND | PURGE]
[, EXTENTS (primary, secondary, maximum)]
[, MAXFILES num_files]
[, MEGABYTES number]
[, OWNER (group_number, user_number)]
[, SECURE "[, FORMAT RELEASE major.minor] rwep"]
```

FORMAT RELEASE major.minor

Specifies the trail format of the data that is sent by Extract to a trail, file, or (if a remote task). The value tells the reader process whether the data records are of a version that it supports. The format depends on the version of the Oracle GoldenGate process. Newer Oracle GoldenGate versions contain different characteristic than older ones.

- FORMAT is a required keyword
- RELEASE specifies an Oracle GoldenGate release version. major is the major version number, and minor is the minor version number. The X.x must reflect a current or earlier, generally available (GA) release of Oracle GoldenGate. Valid values are 9.0 through the current Oracle GoldenGate X.x version number, for example 11.2 or 12.1. (If you use an Oracle GoldenGate version that is earlier than 9.0, specify either 9.0 or 9.5.) The release version is programmatically mapped back to an appropriate internal compatibility level. The default is the current version of the process that writes to this trail.

file_name
A physical file name or an existing define name of class map.

APPEND | PURGE

- Specify `APPEND` to append to the file.
- Specify `PURGE` to purge the file.

EXTENTS (*primary*, *secondary*, *maximum*)
Sets up the extent sizes and maximum extents for the file.

MAXFILES *num_files*
Valid for Extract. Forces a sequence of files to be created, rather than a single file. `MAXFILES` permits up to *num_files* to be created as needed. Aged files are appended with a six-digit sequence number. When using `MAXFILES`, `MEGABYTES` should also be specified in order to explicitly set the maximum size of each file in the sequence.

MEGABYTES *number*
Sets up the maximum size of the file, or sequence of files if you specified `MAXFILES`. The maximum size of each file is 2 gigabytes.

OWNER (*group_number*, *user_number*)
Gives ownership of the file to a different owner. Securing a file is useful when you extract logical sets of records to different files (for example, accounting data to one and sales data to another), and only want the sales group to have access to sales information.

SECURE "*rwep*"
Secures the file using standard Guardian security.

Example

The following example illustrates how the `EXTFILE` parameters work in conjunction with the file and table parameters. This example extracts data owned by the `EAST` region into one file and data owned by `WEST` into another.

```
EXTFILE $DATA1.EXTRACT.EAST, OWNER (100,255),
SECURE "GGGG", PURGE
        TABLE $DATA5.SALES.ORDERS WHERE (REGION = "EAST");
        TABLE $DATA5.ACCTING.RECEIPTS WHERE (REG = "E");
EXTFILE $DATA1.EXTRACT.WEST, OWNER (200,255),
SECURE "GGGG", PURGE
        TABLE $DATA5.SALES.ORDERS WHERE (REGION = "WEST");
        TABLE $DATA5.ACCTING.RECEIPTS WHERE (REG = "W");
```

EXTRACT

Valid for

Extract

Description

Use `EXTRACT` to associate with the Extract group defined by the GGSCI `ADD EXTRACT` command. The association with the group ensures that each extracted record is processed exactly once.

One of `EXTRACT`, `SPECIALRUN` or `SOURCEISFILE` entry is required.

Syntax

```
EXTRACT group_name
```

group_name
The name of the group. Extract *group_name* can contain no more than 7 characters. It must be the first parameter in the Extract parameter file.

EXTTRAIL

Valid for

Extract, Replicat

Description

Use `EXTTRAIL` to establish the current Oracle GoldenGate trail to which data will be output. All `FILE` and `TABLE` entries after the current entry but before the next parameter (`EXTFILE`, `EXTTRAIL`, `RMTBATCH`, `RMTFILE`, `RMTTRAIL`) result in output to the current trail. Unlike `EXTFILE`, `EXTTRAIL` parameters are set up externally to the parameter file using GGSCI.

The trail entered must correspond with an Oracle GoldenGate trail created with GGSCI. In addition, the Extract group specified in the parameter file must match the entry linked with `EXTTRAIL`. For more information about adding Oracle GoldenGate trails, refer to "ADD ".

Syntax

```
EXTTRAIL file_[, FORMAT RELEASE major.minor]prefix
```

FORMAT RELEASE major.minor

Specifies the trail format of the data that is sent by Extract to a trail, file, or (if a remote task). The value tells the reader process whether the data records are of a version that it supports. The format depends on the version of the Oracle GoldenGate process. Newer Oracle GoldenGate versions contain different characteristic than older ones.

- FORMAT is a required keyword.
- RELEASE specifies an Oracle GoldenGate release version. major is the major version number, and minor is the minor version number. The X.x must reflect a

current or earlier, generally available (GA) release of Oracle GoldenGate. Valid values are 9.0 through the current Oracle GoldenGate X.x version number, for example 11.2 or 12.1. (If you use an Oracle GoldenGate version that is earlier than 9.0, specify either 9.0 or 9.5.). The release version is programmatically mapped back to an appropriate internal compatibility level. The default is the current version of the process that writes to this trail.

file_prefix
A file name with two characters in the file portion. *file_prefix* can also be a define with the same characteristics.

Example

An Extract group, FINANCE, has two associated trails. This configuration sends ACCOUNTS records to the XX trail and ORDERS to the YY trail. The parameter file includes the following commands:

```
EXTRACT FINANCE
...
EXTTRAIL $DATA1.EXTDAT.XX
FILE $DATA2.FINANCE.ACCOUNTS;
EXTTRAIL $DATA2.EXTDAT.YY
FILE $DATA3.FINANCE.ORDERS;
```

FASTIO

Valid for

Extract

Description

Use FASTIO to output records in large blocks of up to 28K bytes, resulting in high performance gains. Use FASTIO in high volume scenarios. It is less important when data extract rates fall below several hundred megabytes per hour.

> **Note:**
>
> FASTIO only applies to EXTTRAIL, not to RMTTRAIL.

Syntax

```
FASTIO
```

FASTPOSITION | NOFASTPOSITION

Valid for

Extract

Description

Use FASTPOSITION to instruct the Audserv program to perform a binary search of the TMF audit trail at startup, before the initial checkpoint is established. This significantly reduces the startup time and CPU overhead associated with starting the process for the first time. This parameter is particularly useful for systems that have implemented auxiliary TMF audit trails.

FASTPOSITION does not apply to Oracle GoldenGate trails, SOURCEISFILE, or direct file reads.

Default

FASTPOSITION

Syntax

FASTPOSITION

FASTREADS | NOFASTREADS

Valid for

Coordinator, Extract, Replicat

Description

Use FASTREADS to change the number of bytes that Extract reads when processing Oracle GoldenGate trails. Extract reads up to 4096 bytes at a time by default. FASTREADS enables larger reads of up to 28K bytes. When data volumes are significant, FASTREADS can result in greater throughput and lower overhead.

> **Note:**
>
> FASTREADS only applies when Extract is reading an Oracle GoldenGate trail and does not apply when reading TMF audit trails, since Oracle GoldenGate always reads audit trails with a large block read.

Default

NOFASTREADS

Syntax

FASTREADS

FETCHCOMPS | FETCHLASTIMAGE

Valid for

Extract

Description

Use FETCHCOMPS or FETCHLASTIMAGE to extract full update images. When audit compression is used for a SQL table or Enscribe file, update operations to the table or file are recorded in a compressed format. Therefore, only part of the image is available for updates. (Full delete and insert images are always available). For some applications, this is acceptable. For example, when delivering the operation to another file, usually you need only the portion of the record that changed.

When either FETCHCOMPS or FETCHLASTIMAGE is specified, Extract attempts to retrieve the full record images of any compressed records from the original database using a SQL SELECT or Guardian READ statement.

You can fetch compressed images for selected files or tables. FETCHCOMPS and FETCHLASTIMAGE apply only to FILE or TABLE entries listed below the FETCHCOMPS or FETCHLASTIMAGE specification. Specify NOFETCHCOMPS or NOFETCHLASTIMAGE to turn off fetching for subsequent entries.

FETCHCOMP is more useful for audit applications, and FETCHLASTIMAGE is more appropriate for hot site applications. Several other items to note when using these parameters:

- FETCHLASTIMAGE outputs the latest image of the record found on disk. Changes that occurred during the current transaction are ignored.
- FETCHCOMPS uses data from the latest image to fill in missing values in the compressed record. This approximates but is not always exactly the same as the actual transaction values (if subsequent updates occurred).
- If the original record has been deleted from the database, Extract discards the record from the compressed record with a warning message.
- The record retrieved may be a newer version than the record processed from the audit trail, so intermediate update information will be lost.
- Fetching each update from the database can result in significant performance penalties because each record must be fetched with random access into the database (Extract processes other updates sequentially).

Syntax

```
FETCHCOMPS | FETCHLASTIMAGE | NOFETCHCOMPS | NOFETCHLASTIMAGE
```

FILE

Valid for

Logger

Description

Use FILE to specify file sets and record types to the current log. You can also use the EXCLUDEFILE parameter to subsequently exclude files that have been included by FILE.

You can include files in the current log by:

- Specifying file sets
- Specifying filtering values

- Optionally, you can specify a variety of attributes to enhance file processing.

Syntax

```
FILE file_set [, attribute...][, filter];
```

file_set
The name of the file, or file set to be processed.

attribute
For details about the attributes you can specify as `FILE` options see "Using parameters as FILE options".

filter
Can be one or more of the following:

- `PROCESS process_file_set`
- `PROGRAM program_file_set`
- `USER user_id`

If more than one filter is specified, *all* of the specified criteria must be met. See "Including with filters" for details about using filters.

Specifying file sets

`FILE` can explicitly name a file set, as in `$DATA3.DAT.TRANS`, or specify wildcard arguments as in `$DATA4.*.*`.

When multiple `FILE` statements are included in a parameter file, only the first match is logged. The following is an example:

```
FILE $DATA4.GGSDAT.FIN*
FILE $DATA4.GGSDAT.*
```

In this case, the file set `$DATA4.GGSDAT.FIN*` is logged only once, to the corresponding log, even if a subsequent `FILE` (such as `$DATA4.GGSDAT.*`) implicitly includes it in a wildcard argument.

The primary partition of the file must be satisfied by a `FILE` entry for extraction to take place. Also, excluding secondary partitions has no effect.

If you wish to retrieve a file set from a remote node, Logger must be configured and running on that node locally.

Using parameters as FILE options

You can use other parameters as options with `FILE`; this lets you apply a parameter to a single file or file set, rather than all the files in your Logger group. Parameters that can be set as an option in `FILE` include:

- `COMPRESSUPDATES | NOCOMPRESSUPDATES` (default)
- `GETAUDITED | IGNOREAUDITED` (default)
- `GETBEFOREUPDATES | IGNOREBEFOREUPDATES` (default)
- `GETBULKIO | IGNOREBULKIO` (default)
- `GETUNSTRUCTURED | IGNOREUNSTRUCTURED` (default)
- `RENAMEDELAY delay_seconds`

- `SUSPENDED | ACTIVE` (default)

Including with filters

A file can also be included based on filters. If you specify more than one filter, *all* of the filter criteria must be met to include the file for logging. For example, the following only includes the specified file set when both the process and user ID filter criteria are met.

```
FILE $DATA3.APPL.TL*, PROCESS $APP*, USER SUPER.*
```

If a file is included by `FILE` without filters, then excluded according to one or more filters, the file is excluded when the filter criteria are met.

Syntax

```
FILE file_set
[, PROCESS process_file_set]
[, PROGRAM program_file_set]
[, USER user_id]
;
```

`PROCESS` *`process_file_set`*
Includes files sets that are opened by the specified process.

`PROGRAM` *`program_file_set`*
Includes file sets that are opened by the specified program.

`USER` *`user_id`*
Includes file sets when the creator ID of the opening process or program is associated with the specified user ID.

Specifying attributes

You can optionally specify attributes for:

- Compressing updates
- Capturing or omitting record types
- Supporting file renaming
- Suspending and resuming logging

To specify multiple attributes, include each attribute in a separate `FILE` statement, similar to:

```
COMPRESSUPDATES
FILE $DATA1.DAT.PAYMENT, NOCOMPRESSUPDATES
FILE $DATA1.DAT.ACCOUNT, GETBULKIO
FILE $DATA1.DAT.TRANS, RENAMEDELAY 10
```

This example does the following:

- Compresses updates for subsequent `FILE` statements.
- Turns compression off for the `PAYMENT` file.
- Logs bulk I/O for the `ACCOUNT` file.
- Supports file renaming by delaying the actual rename of the `TRANS` file until the name is changed in the log.

See the following subsections for details about using the `FILE` parameter attributes.

Compressing updates

You can compress and decompress update records using the COMPRESSUPDATES and NOCOMPRESSUPDATES options.

By default, update records are not compressed. The default can result in lower throughput, especially across Wide Area Networks, due to the additional traffic load. For example, consider an application that updates a 1000-byte customer record with a key 20 bytes long. If typically only the balance field within the customer record is changed, and that field is 10 bytes long, only 38 bytes rather than 1000 need to be transmitted across the network to execute a replicated update (20 bytes for the key, 10 bytes for the balance field, and 8 bytes to indicate the position within the record of the changed bytes and the key).

If you wish to use a field in KEYCOLS which is not part of the source file's primary key, you cannot use COMPRESSUPDATES.

Syntax

```
FILE file_set, {COMPRESSUPDATES | NOCOMPRESSUPDATES}
```

NOCOMPRESSUPDATES
No compression (the default).

COMPRESSUPDATES
COMPRESSUPDATES directs Logger to compress update record images by comparing before and after-images. Compressed images include the key of the changed record and only the changed bytes within the record. Note that if you wish to use a field in KEYCOLS which is not part of the source file's primary key, you cannot use COMPRESSUPDATES.

Capturing or omitting record types

You can specify that Logger include or omit the following record types:

- TMF audited records
- Before updates
- Bulk I/O updates
- Unstructured file changes
- Omit inserts, updates, or deletes.

Syntax

```
FILE file_set
[, GETAUDITED | IGNOREAUDITED]
[, GETBEFOREUPDATES | IGNOREBEFOREUPDATES]
[, GETBULKIO | IGNOREBULKIO]
[, GETUNSTRUCTURED | IGNOREUNSTRUCTURED]
[, OMITINSERTS | OMITUPDATES | OMITDELETES] ;
```

GETAUDITED | IGNOREAUDITED
Retrieves or omits data from files that are TMF audited.

> **Note:**
>
> Carefully consider possible outcomes before using GETAUDITED. Logger cannot capture TMF abort operations. When GETAUDITED is used, Logger will capture operations that it will not be able to back out if TMF aborts them.

GETBEFOREUPDATES | IGNOREBEFOREUPDATES
Retrieves or omits the before-images of records.
GETBEFOREUPDATES retrieves images of records before they are changed in addition to capturing the after record image. For example, if an account balance was $100 before a transaction, and after it was $1000, both records will be written to the log trail. This information can be useful for data warehousing, archival, and other applications. When records are deleted, before-images are always logged. The default is to not extract before-images.

GETBULKIO | IGNOREBULKIO
Retrieves or omits bulk I/O updates. IGNOREBULKIO (omit bulk I/O updates) is the default. Bulk I/O occurs when Guardian SETMODE 141 is invoked by a program, such as during FUP LOAD or FUP DUP.

GETUNSTRUCTURED | IGNOREUNSTRUCTURED
Retrieves or omits unstructured file changes. IGNOREUNSTRUCTURED (omit unstructured file changes) is the default.

OMITINSERTS | OMITUPDATES | OMITDELETES
Excludes the specified record operation from being captured. You can combine these options.

Supporting file renaming

If your application renames database files while they are still open and being updated, the RENAMEDELAY option ensures that the new file name changes in the log trail. For RENAMEDELAY, specify a delay interval to give the system time to detect and record the new name.

Syntax

```
FILE file_set, RENAMEDELAY delay_seconds
```

delay_seconds
Represents the delay interval, as in RENAMEDELAY 10. The interval should range from 5-15 seconds, depending on overall system resource usage and hardware capacity. In general, more powerful systems require less delay. Use this feature with caution, since the process invoking the rename will be delayed before being allowed to continue.

Suspending and resuming logging

You can temporarily suspend, then resume logging for a specified file set by using the SUSPENDED and ACTIVE options.

Syntax

```
FILE file_set, {SUSPENDED | ACTIVE} ;
```

SUSPENDED
Temporarily suspends logging for the particular file set.

ACTIVE
Resumes logging for the file set.

FILE | TABLE

Valid for

Extract

Description

Use `FILE` or `TABLE` to specify the files or tables for which to capture data. You can specify a file name, or a wildcard arguments such as `$DATA3.*.*`. If you are retrieving records from remote locations, you must fully qualify the file name with its node as well as the volume, subvolume, and file. (For simplicity, references to `FILE` in this section also refer to `TABLE` unless explicitly stated otherwise.)

For Enscribe, unless you specify otherwise, records from every partition of the specified file are retrieved.

You can invoke `FILE` or `TABLE` more than once in a parameter file, and you can invoke the same `FILE` or `TABLE` argument more than once. This is useful, for example, to split records into different trails according to column values, to put inserts, updates and deletes into separate files, and to segment data for other reasons.

> **Note:**
>
> At least one `FILE` or `TABLE` statement per parameter file is required.

Syntax

```
FILE file_name
[, ALTNAME alternate_file_name]
[, AUTOTRUNCATE]
[, COLMAP (column_map_specification) | NOCOLMAP]
[, COMPRESSDELETES]
[, DEF source_ddl_definition]
[, EVENTACTIONS (action_options)
[, EXITPARAM "exitparam_string"]
[, FILTER (expression)]
[, KEYCOLS (key_column_specification)]
[, PARTITIONS partition_specification]
[, RANGE (range_specification)]
[, SQLEXEC (sqlexec_clause)]
[, SQLNAME table_alias]
[, STARTKEY key_specification, ENDKEY key_specification]
[, TARGET target_file_name]
[, TARGETDEF target_ddl_definition]
[, TARGETNAME target_file_name]
[, USETARGETDEFLENGTH]
[, TOKENS (token_specification)]
```

```
[, WHERE (where_condition)]
;
```

file_name
A physical file name or an existing define name of `CLASS MAP` or a wildcard file name. The file can be a SQL table, SQL view or Enscribe file

ALTNAME
See "Handling missing files".

AUTOTRUNCATE
See "Purging Records for Initial Load".

COMPRESSDELETES
See "Compressing Records".

COLMAP SQLNAME TARGETDEF TARGETNAME USETARGETDEFLENGTH
See "Mapping Data".

DEF FILTER PARTITIONS STARTKEY, ENDKEY RANGE WHERE
See "Selecting Records".

EVENTACTIONS (*action_options)*
Se "Triggering Actions".

EXITPARAMS
See "Passing literal strings to user exits".

SQLNAME
See "Specifying a table alias".

SQLEXEC
See "Performing a query".

TOKENS
See "Using tokens".

Compressing Records

Use `COMPRESSDELETES` to replicate only the primary keys for deleted records. Without this parameter, all columns are replicated. By sending only the primary key, Oracle GoldenGate has all of the data required to delete the target record, while restricting the amount of data that must be processed.

Syntax

```
FILE file_name, COMPRESSDELETES
```

Selecting Records

You can select records by:

- Selecting or excluding records using `FILTER`.
- Selecting based on a conditional statement using `WHERE`.
- Selecting a subset of records using `RANGE`.
- Selecting a specific data partition using `PARTITIONS`.

- Selecting Enscribe records based on a STARTKEY and ENDKEY.

> **Note:**
>
> Using the RANGE option of FILE or MAP provides different capabilities than using the @RANGE function within a FILTER. And both of these are different than the RANGE option of ALTINPUT.

Selecting or Excluding Records Using FILTER

In the FILTER expression, records are selected according to a filter clause. Options specify the record types to include or omit when applying the filter. You can combine the filter clause with one or more options, but the *filter_clause* must always be included.

If you are selecting from an Enscribe file using FILTER, you must also specify the DEF option.

Syntax

```
FILE file_name,
FILTER (filter_clause
[, ON INSERT | ON UPDATE| ON DELETE]
[, IGNORE INSERT | IGNORE UPDATE | IGNORE DELETE])
[, DEF source_ddl_definition]
;
```

ON INSERT | ON UPDATE | ON DELETE
Include in the filter expression to specifically limit the filter clause to be executed on an insert, update or delete. You can specify more than one option. For example, ON UPDATE, ON DELETE executes on updates and deletes, but not inserts.

IGNORE INSERT | IGNORE UPDATE | IGNORE INSERT
Ignores the specified operation. You can specify more than one IGNORE option.

DEF *source_ddl_definition*
Has meaning only for Enscribe files. Use a DDL definition or record within the open dictionary. This definition describes the record that is extracted from the TMF audit trails. You cannot specify more than one definition for any FILE statement.

Selecting Based on a Conditional Statement

With the WHERE option, you can select information based on a conditional statement. If you are selecting from an Enscribe file using WHERE, you must also specify the DEF option.

Syntax

```
FILE file_name, WHERE (where_condition)
[, DEF source_ddl_definition];
```

where_condition
Selects a subset of records from a source file or table, based on a condition, such as WHERE (branch = "NY"). For a list of valid operators, see Table 2-32.

DEF *source_ddl_definition*
Has meaning only for Enscribe files. Use a DDL definition or record within the open dictionary. This definition describes the record that is extracted from the TMF audit trails. You cannot specify more than one definition for any FILE statement.

Table 2-32 Permissible WHERE operators

Operator	Example
Column names	PRODUCT_AMT
Numeric values	-123, 5500.123
Literal strings enclosed in quotes	"AUTO", "Ca"
Column tests	@NULL, @PRESENT, @ABSENT (column is null, present or absent in the record). These tests are built into Oracle GoldenGate.
Comparison operators	=, <>, >, <, >=, <=
Conjunctive operators	AND, OR
Grouping parentheses	Use open and close parentheses for logical grouping of multiple elements.

Selecting a Subset of Records Using RANGE

Use the RANGE clause to select a subset of the records in the source file or table. Unlike WHERE, RANGE does not require knowledge of the table or file structure.

Syntax

```
FILE file_name, RANGE (x [, x, ...] OF y);
```

(x [, x,...] OF y)
Selects a subset of records from a source file or table, based on a condition, such as RANGE (3 of 5) or RANGE (1, 3 of 5).

Duplicate unique index errors are possible when using the RANGE option for a file with a unique alternate key. For example, you delete the primary key for a record with primary key A and alternate key B. Then you insert with a different primary key but the same alternate key (e.g. primary key C, alternate key B). RANGE separates records based on the primary key, so transaction A and C can be sent to different Replicats and encounter a duplicate unique index on B when the insert is picked up first.

To avoid this, use the FILTER (@RANGE) function explained on "RANGE"and supply the columns that make up the unique alternate key as illustrated in the following example.

Example

In this example table TAB1 has three columns. The first two, COL1 and COL2, form the primary key and the third is the unique alternate index named U_INDX_COL. The range for the trail files is set up to override the primary key with the unique index column as shown below.

```
EXTTRAIL AA
TABLE TAB1, FILTER (@RANGE (1, 2, U_INDX_COL));
EXTTRAIL BB
TABLE TAB1, FILTER (@RANGE (2, 2, U_INDX_COL));
```

Selecting a Specific Data Partition

Use the PARTITIONS option to specify which partitions of the file or table to write to the current Oracle GoldenGate trail. Particular partitions can be output to specific files in the trail, or skipped altogether.

Use PARTITIONS only when you have specified SOURCEISFILE and either FASTUNLOAD or FASTUNLOADSHARED. Otherwise, PARTITIONS has no effect.

Syntax

```
FILE file_name, PARTITIONS (volume_specification);
```

volume_specification
A volume name, a volume number, or a range of volume numbers. Volume numbers begin with zero, and the last volume number can be specified as L. You can specify multiple volumes, delimited by commas, as in the following examples:

```
TABLE XYZ, PARTITIONS
(\LA.$DATA1, $DATA3, \XYZ.$SYSTEM);
TABLE ABC, PARTITIONS
(0, 2 - 5, 10 - L);
```

Selecting Enscribe Records Based on Key

STARTKEY and ENDKEY can be used to limit the range of the keys that will be selected if your parameter settings meet the following requirements:

- The FILE statement must specify an Enscribe file as the source.
- SOURCEISFILE must apply.
- Either SOURCEDEFS or both DICTIONARY and DEF must be present.
- If SOURCEDEFS is used, the access cannot be by ALTKEY.
- The PARTITIONS option cannot be used.
- The FASTUNLOAD option cannot be used.

The columns used in the key specification must make up the high order portion of a system key, primary key, or alternate key defined for the Enscribe file. A SOURCEDEFS or DICTIONARY must be present to define the column name and value.

Both STARTKEY and ENDKEY are required and both are evaluated when deciding whether to select the record. Any existing FILTER or WHERE clauses are processed for records selected based on key range.

STARTKEY and ENDKEY can be defined for different FILE statements in the same parameter file.

Syntax

```
FILE file_name
[, STARTKEY key_specification, ENDKEY key_specification]
```

Example

```
FILE $NY.ACCT.MASTER, STARTKEY (DIV="A1", ACCTNO=00000),
                      ENDKEY (DIV="Z9", ACCTNO=49999),
                      DEF $NY.DDLDEF.ACTDEF;
```

Mapping Data

Oracle GoldenGate has the following data mapping capability for the `FILE` or `TABLE` parameters:

- Mapping columns.
- Retrieving data layout during Replicat processing.
- Invoking a user exit.

Mapping Columns

Using a `COLMAP` clause, you can extract fields or columns from one record and map them to a differently structured record. This is useful, for example, when delivering data from an Enscribe file to an SQL table with similar, but not identical, fields. `COLMAP` selects, translates, and moves the fields you want into the new structure. When associated records are output, they are identified by the target file rather than the source to reflect the new structure of the record.

- When mapping from an Enscribe file, include either the `DEF` or the `TARGETDEF` option. Use `DEF` when capturing and mapping from the trail. If the target is an Enscribe file, you must associate a DDL definition with the target so that mapping instructions can be interpreted with `TARGETDEF`.
- When SQL tables are identified as the target, the layout of the target record after mapping is known since the SQL table structure is retrieved from the SQL catalog.

Additionally, you can match the source record length to the target record length. See "Matching Source and Target Record Lengths".

`COLMAP` requires the `TARGET` option. A DDL definition for an Enscribe target is required only once in the parameter file, and only when using a `COLMAP` clause.

Syntax

```
FILE file_name,
COLMAP (column_map_specification) | NOCOLMAP
TARGET target_file_name
[, DEF source_ddl_definition | TARGETDEF target_ddl_definition]
;
```

column_map_specification
The column mapping expression, as in:

```
(target_column = source_expression)
```

Explicitly defines a source-target column map where *target_column* is the name of the target column and *source_expression* can be any of the following:

- Numeric constant, such as 123
- String constant enclosed within quotes, such as "ABCD"

- The name of a source column, such as `ORD_DATE`
- An expression using an Oracle GoldenGate column-conversion function, such as `@STREXT (COL1, 1, 3)`

NOCOLMAP
Allows the user to specify a `DEF` for filtering purposes, but prevents the column-mapping command from completing. Example:

```
MAP \PROD.$DATA06.CER1ATLF.TL*,
TARGET \GGS2.$DATA10.GGSLOGS.*,
DEF TLF,
NOCOLMAP,
WHERE (TLF.HEAD.REC-TYP <> "00");
```

TARGET *target_file_name*
Names the target file. Must be an existing Enscribe file or SQL table, and can be an active define name.

DEF *source_ddl_definition*
Has meaning only for Enscribe files. Use a DDL definition or record within the open dictionary. This definition describes the record that is extracted from the TMF audit trails. You cannot specify more than one definition for any `FILE` statement.

TARGETDEF *target_ddl_definition*
Use `TARGETDEF` when invoking column mapping to an Enscribe file structure and the Enscribe file has not yet been specified in the parameter file, or did not have a definition associated with it.
If you assigned a definition to the target file earlier in the parameter file, you can omit `TARGETDEF`.

Matching Source and Target Record Lengths

Use the `USETARGETDEFLENGTH` option to adjust the source record length to the length of the target record. Precede the `COLMAP` statement with the `USETARGETDEFLENGTH` option.

Syntax

```
USETARGETDEFLENGTH
USETARGETDEFLENGTH
COLMAP (USEDEFAULTS,
CRD-TYP = @STRSUB (CRD-TYP, "VD", "PV"),
FIID = @STRSUB (FIID, "WA", "SFNB"),
FIID = @STRSUB (FIID, "ID", "BAID"));
```

Retrieving Data Layout during Replicat Processing.

The table name stored in the remote trail can be different for each record. Using the `TARGETNAME` option, you can specify a different file name in the header for each record retrieved from the file. Replicat uses the new file name to resolve the file or table layout.

If the specified file exists at the target node, Replicat can retrieve the layout from a local catalog or dictionary, which can save a significant amount of time.

Syntax

```
FILE file_name, TARGETNAME target_file_name;
```

target_file_name
The name of the target file for which Replicat retrieves the layout.

Triggering Actions

`EVENTACTIONS` can be used to trigger an event based on a file or table receiving a DML record.

Syntax

```
FILE source_file
EVENTACTIONS ([VERBOSE]
  [, ROLLOVER]
  [, IGNORE | DISCARD]
  [, TACLCMD ("CMD_file_details") | TACLCMD (CMD $1, VAR $1 = value)]
  [, EMS WARN | INFO]
  [, CHECKPOINT {BEFORE | AFTER | BOTH}]
  [, REPORT]
  [, STOP]
  [, SUSPEND])
```

VERBOSE
Writes details to the report for each event as it is processed

ROLLOVER
Increments the sequence number of the output trail.

IGNORE
Skips the record. `IGNORE` and `DISCARD` are incompatible, so only one of these options can be used.

DISCARD
Discards the record. `IGNORE` and `DISCARD` are incompatible, so only one of these options can be used.

TACLCMD
Executes a user-defined system command. Valid `TACLCMD` file commands are `PURGE`, `PURGEDATA`, `RUN`, `RENAME`, `OBEY`, and `FUP`. Non-file values can be obtained by `SQLEXEC`, `@GETENV()`, or Column Data.

EMS
Writes either `INFO` or `WARN` messages to EMS.

CHECKPOINT
Checkpoints its position `BEFORE` the record is processed, `AFTER` the record is processed or `BOTH`.

REPORT
Writes the current statistics to the report file.

STOP
Stops the process.

SUSPEND
Suspends the process until it is resumed by a command from GGSCI.

```
FILE $DATA.SLS.PLR,
  EVENTACTIONS (VERBOSE, TACLCMD "RENAME $DATA.SLS.PLR, $DATA.SLS.T4M");
```

Example

The following example writes to the report and executes a TACLCMD to rename the file.

Purging Records for Initial Load

When extracting data for an initial load, you can use the AUTOTRUNCATE option to send a PURGEDATA record to the trail as the first record. The PURGEDATA purges the target file before Replicat applies any data, so that the target file is loaded from a clean state.

Use AUTOTRUNCATE with extreme caution, since it causes all existing data to be purged from the target file.

- Do not use AUTOTRUNCATE if you are performing multiple direct loads to the same target file, such as when using a range function to distribute the processing load.
- AUTOTRUNCATE is not suited to a bi-directional configuration. PURGEDATA is a DDL statement that is automatically committed and not linked to any transaction, making loop detection difficult. Without effective loop detection, AUTOTRUNCATE could cause not only target, but also original source files, to be purged of data. If you use AUTOTRUNCATE in a bi-directional configuration, you should use IGNOREPURGEDATAS in your online Extract groups.

Handling missing files

Use the ALTNAME option to specify an alternate name for a file that may no longer exist. Many applications use daily transaction files; purging the previous file and creating a new transaction file each day. A problem can occur when Extract expects to process data from a purged or renamed file.

Using ALTNAME, you can specify a generic definition for these types of files. This requires that a generic file exist that accurately describes the structure of each of the files in a group. The generic file does not require any data.

This option applies only when the source is an Oracle GoldenGate or Logger trail.

Syntax

```
FILE file_name, ALTNAME alternate_file_name
[, DEF source_ddl_definition];
```

alternate_file_name
The alternate file name.

DEF *source_ddl_definition*
Has meaning only for Enscribe files. Use a DDL definition or record within the open dictionary. This definition describes the record that is extracted from the TMF audit trails. You cannot specify more than one definition for any FILE statement.

Example

```
FILE $DATA1.DAT.TR*, ALTNAME $DATA1.TEMPLATE.TRANS, DEF TRANS-DEF;
```

Passing literal strings to user exits

Use EXITPARAM "*exitparam_string*" to pass a literal string to user exit routines whenever a record from FILE is encountered.

The string must be enclosed in double quotes and an ampersand used if it continues to additional lines. It is unlimited in size, but you must use the new function `GET_EXIT_PARAM_VALUE` to access values over the default of 256 bytes.

Syntax

```
FILE file_name, EXITPARAM "exitparam_string";
```

Specifying a table alias

When you specify the `FORMATASCII`, `FORMATSQL, or FORMATXML` parameters, you can use `SQLNAME` to substitute a string for the table name in the output. To preserve lowercase attributes of the string, enclose the string in quotes.

Syntax

```
FILE file_name, SQLNAME table_alias;
```

Performing a query

Use `SQLEXEC` to perform a SQL query when processing a record for a SQL/MP table. `SQLEXEC` enables Oracle GoldenGate to communicate directly with the database to perform any query that SQL supports. The database function can be part of the synchronization process, such as retrieving values for column conversion, or it can be independent of extracting or replicating data.

> **Note:**
>
> This feature is not available for Enscribe files. `SQLEXEC` queries should not be used to change a value in the primary key column. The primary key value is passed from Extract to Replicat, so Replicat can perform further update/delete operations. If Replicat does not know the primary key value, these operations cannot be completed.

By using `SQLEXEC` within multiple `FILE` or `MAP` statements, you can create different rules for different tables; these rules can be as simple or as complex as needed. A query that is executed can accept input parameters from source or target rows and pass output parameters.

In the following example, `SQLEXEC` runs a select statement, extracting the timestamp from the target table, then filters out records as needed.

```
FILE $DATA1.SQLDAT.ORDERS,
SQLEXEC (ID check,
QUERY " SELECT TIMESTAMP FROM $DATA1.SQLDAT.ORDERS "
" WHERE PKCOL =?P1 ", PARAMS (P1 = PKCOL), ERROR REPORT);
```

A `SQLEXEC` statement expects legal SQL syntax for the database being affected. Refer to the SQL for NonStop reference guide for permissible SQL syntax.

> **Note:**
>
> If a SQLEXEC query fails, the Extract or Replicat process will exit. As such, you must structure your query correctly.

Syntax

```
FILE file_name, SQLEXEC (
ID logical_name,
QUERY "sql_query",
{PARAMS param_spec | NOPARAMS}
[, AFTERFILTER | BEFOREFILTER]
[, DBOP]
[, EXEC frequency]
[, MAXVARCHARLEN bytes]
[, PARAMBUFSIZE num_bytes]
[, TRACE option]
[, ALLPARAMS option]
[, ERROR action]
[, ...]
)
```

ID *logical_name*
Defines a logical name for the query. A logical name is required in order to extract values from the query results. ID *logical_name* references the column values returned by the query.

QUERY "*sql_query*"
Specifies the SQL query syntax to execute against the database. The query must be valid, standard query statement for the database against which it is being executed. It can either return results with a SELECT statement or update the database with an INSERT, UPDATE, or DELETE statement.
For any query that produces output with a SELECT statement, only the first row returned by the SELECT is processed. Do not specify an "INTO..." clause for any SELECT statements.
Enclose the query within quotes. For a multi-line query, use quotes on each line. To ensure success, place a space after each begin quote and each end quote, or at least before the end quote.
For example, in the following, there are spaces before the words select and where and after the words ggs_notify and ?p1."

```
SQLEXEC (
ID ggs, ON UPDATES, ON INSERTS,
QUERY " select notified from $DATA1.SQLDAT.NOTIFY "
" where account_no = ?p1 ",
PARAMS (p1 = account_no)
)
```

PARAMS *param_spec* | NOPARAMS
Defines whether the query accepts parameters. One of these options must be used.

AFTERFILTER | BEFOREFILTER
Specifies when to execute the query in relation to a FILTER clause. AFTERFILTER executes after the filter and enables you to skip the overhead of executing the SQL

unless the filter is successful. This is the default. `BEFOREFILTER` executes before the filter and enables you to use the results of the procedure or query in the filter.

DBOP
Commits `INSERT`, `UPDATE`, `DELETE`, and `SELECT` statements executed within the query. Otherwise, they could potentially be rolled back. Oracle GoldenGate issues the commit within the same transaction boundaries as the source transaction. Use caution: any changes that are committed by the procedure can result in overwriting existing data.

EXEC *frequency*
Controls the frequency with which a query executes and how long the results are considered valid, if extracting output parameters. Takes one of the following arguments:

MAP
Executes the query once for each source-target table map for which it is specified. `MAP` renders the results invalid for any subsequent maps that have the same source table. For example, if a source table is being synchronized with more than one target table, the results would only be valid for the first source-target map. `MAP` is the default.

ONCE
Executes the query once during the course of an Oracle GoldenGate run, upon the first invocation of the associated `FILE` or `MAP` statement. The results remain valid for as long as the process remains running.

TRANSACTION
Executes the query once per source transaction. The results remain valid for all operations of the transaction.

SOURCEROW
Executes the query once per source row operation. Use this option when you are synchronizing a source table with more than one target table, so that the results of the procedure or query are invoked for each source-target mapping.

MAXVARCHARLEN *bytes*
Specifies the maximum length allocated for any output parameter in a query. Beyond this maximum, output values are truncated. The default is 255 bytes without an explicit `MAXVARCHARLEN` clause.

PARAMBUFSIZE *num_bytes*
Specifies the maximum size of the memory buffer that stores parameter information, including both input and output parameters. Oracle GoldenGate issues a warning whenever the memory allocated for parameters is within 500 bytes of the maximum. The default is 10,000 bytes without an explicit `PARAMBUFSIZE` clause.

TRACE *option*
Takes one of the following arguments:

TRACE ALL
Writes the input and output parameters of each invocation of a query to the report file.

TRACE ERROR
Writes parameters to the report file only if an error occurs.

ALLPARAMS *option*
Takes one of the following arguments:

ALLPARAMS REQUIRED
Indicates that all parameters must be present for the queries to execute.

ALLPARAMS OPTIONAL
Allows the query to execute without all parameters being present.

ERROR *action*
Requires one of the following arguments:

ERROR IGNORE
Database error is ignored and processing continues.

ERROR REPORT
Database error is written to a report.

ERROR RAISE
Database error is handled just as a table replication error.

ERROR FINAL
Database error is handled as a table replication error, but does not process any additional queries.

ERROR FATAL
Database processing abends.

Using tokens

Use `TOKENS` to define a user token and associate it with data. Tokens enable you to extract and store data within the user token area of a trail record header. Token data can be retrieved and used in many ways to customize the delivery of Oracle GoldenGate data. For example, you can use token data in column maps or macros.

To use the defined token data in target tables, use the `@TOKEN` column-conversion function in the `COLMAP` clause of a Replicat `MAP` statement. The `@TOKEN` function maps the name of a token to a target column.

Syntax

```
FILE file_name,
TOKENS (token_name = token_data [, ...])
;
```

token_name
A name of your choice for the token. It can be any number of alphanumeric characters and is not case-sensitive.

token_data
A character string of up to 2000 bytes. The data can be either a constant that is enclosed within double quotes or the result of an Oracle GoldenGate column-conversion function.

Example

The following creates tokens named `TK-OSUSER`, `TK-GROUP`, and `TK-HOST` and maps them to token data obtained with the `@GETENV` function.

```
TABLE $DATA.MASTER.ACCOUNT, TOKENS (
TK-OSUSER = @GETENV ("GGENVIRONMENT", "OSUSERNAME"),
TK-GROUP = @GETENV ("GGENVIRONMENT", "GROUPNAME")
TK-HOST =  @GETENV ("GGENVIRONMENT", "HOSTNAME"));
```

FILEAGEDAYS

Valid for

Extract, Replicat

Description

Use `FILEAGEDAYS` this will remove the file/table entries from memory for those that have been resolved from a wildcard and have not been used for a specified amount of time. This should be used in conjunction with `FILEAGESCAN` parameter.

Audserv will inherit this parameter from the Extract.

Default

```
FILEAGEDAYS 2
```

Syntax

```
To check for how long to keep a file not used:
FILEAGEDAYS [n]
FILEAGEHOURS [n]
FILEAGEMINUTES [n]
```

Days: The number of days a file can be inactive before the process will remove it from the file list. Entries can range from 1 to 365 days.

Hours: The number of hours a file can be inactive before the process will remove it from the file list.

Minutes: The number of minutes a file can be inactive before the process will remove it from the file list.

FILEAGESCANDAYS

Valid for

Extract, Replicat

Description

Use `FILEAGESCANDAYS` this will set how often to evaluate `FILEAGE` parameter.

Default

```
FILEAGESCANDAYS 1
```

Syntax

```
FILEAGESCANDAYS [n]
FILEAGESCANHOURS [n]
FILEAGESCANMINUTES [n]
```

Days: The number of days to evaluate the `FILEAGE` parameter. Entries can range from 1 to 365 days.

Hours: The number of hours to evaluate the `FILEAGE` parameter.

Minutes: The number of minutes to evaluate the `FILEAGE` parameter

FILEEXCLUDE

Valid for

Extract

Description

Use `FILEEXCLUDE` to exclude files or file sets from a wildcard list. Specify the `FILEEXCLUDE` parameter after the `EXTFILE` or `EXTTRAIL` entry to which the rule applies. This parameter is an alias for `TABLEEXCLUDE`.

Syntax

```
FILEEXCLUDE file_identifier
```

file_identifier
The file set name or a wildcard argument.

Example

The following example includes file names that begin with `A`, except for files in subvolume `BADSUB`, files that begin with `AB` and the file `$DATA1.BAD.A123`. Data is output into `EXTFILE $DATA1.EXTDAT.EXTFILE1`. Data from any file beginning with `A` would be included in `EXTFILE $DATA1.EXTDAT.EXTFILE2`.

```
EXTFILE $DATA1.EXTDAT.EXTFILE1
FILE $*.*.A;
FILEEXCLUDE $*.BADSUB.*
FILEEXCLUDE $*.*.AB*
FILEEXCLUDE $DATA1.BAD.A123
EXTFILE $DATA1.EXTDAT.EXTFILE2
FILE $*.*.A;
```

FILEOPWARNING

Valid for

Replicat

Description

Use FILEOPWARNING to control Replicat's behavior when it attempts to purge non-existent files. FILEOPWARNING instructs Extract to ABEND, ignore the attempted purge, or issue a warning.

Default

ABEND

Syntax

```
FILEOPWARNING {ABEND | IGNORE | WARN}
```

ABEND
Stops processing

IGNORE
Ignores the purge attempt and continues processing.

WARN
Issues a warning to the report file.

FILERESOLVE

Valid for

Extract

Description

Use FILERESOLVE to alter the rules for processing wildcard file and table entries. By default, wildcard FILE entries are processed each time the wildcard rule is satisfied. If incorrect syntax is entered in the FILE parameter, this can lead to ABEND conditions after startup parameters have been processed.

Replicat also processes file lists with wildcards, but it does so dynamically and the default value of FILERESOLVE DYNAMIC can only be altered for Extract.

Default

DYNAMIC

Syntax

```
FILERESOLVE {DYNAMIC | IMMEDIATE | BOTH}
```

DYNAMIC
Wildcard FILE entries are processed each time the wildcard rule is satisfied.

IMMEDIATE
Existing files or tables that satisfy the wildcard definition are processed at startup.

BOTH
Files existing at startup are processed at that time, and files created after startup are processed when encountered.

Examples

Example 1
Consider the following scenario. $DATA1.DAT.FILE1 exists when Extract starts, and $DATA1.DAT.FILE2 is created after startup.The following example results only in extraction from FILE1. In addition, the list of files to which the wildcard applies is immediately available in the report.

```
FILERESOLVE IMMEDIATE
FILE $DATA1.DAT.*
```

In this case, the file list for $DATA1.DAT.* is resolved immediately, and future additions to this list are ignored.

Example 2
The following example results in the extraction from both TABLE1 and TABLE2. However, the FILE parameter is *not* resolved until a record for TABLE1 or TABLE2 is encountered. Therefore, the MAP statement is not checked for validity until that point.

```
FILERESOLVE DYNAMIC
TABLE $DAT1.DAT.*, TARGET $DATA2.DAT.TEMPLATE,
COLMAP (COL1 = COL2);
```

Example 3
In the following example, if TABLE1 exists at runtime, but TABLE2 does not, the MAP for TABLE1 is checked for syntax immediately, and TABLE2 will still be picked up.

```
FILERESOLVE BOTH
TABLE $DATA1.DAT.*, TARGET $DATA2.DAT.TEMPLATE,
COLMAP (COL1 = COL2);
```

FILLSHORTRECS | NOFILLSHORTRECS

Valid for

Extract, Replicat

Description

Use FILLSHORTRECS to tell the Extract to fill the end of data records to fit their maximum configured length with spaces, zeros or SQL default column values. Use this to generate records in the trails, that are a constant length or contain the same number of columns.

FILLSHORTRECS can also be used for Replicat MAP statements that have a COLMAP statement to trigger mapping.

This functionality can be toggled on and off around FILE or MAP statements by using NOFILLSHORTRECS.

Default

```
NOFILLSHORTRECS
```

Syntax

```
FILLSHORTRECS | NOFILLSHORTRECS
```

FILTERVIEW | NOFILTERVIEW

Valid for

Extract

Description

Use `FILTERVIEW` to process SQL views by entering a view as the object in a `FILE` or `TABLE` entry. When processing views, Extract is actually processing the underlying table in the TMF audit trail. Extract internally maps the table to the view structure.

By default, when processing updates for views, Extract outputs a view update only if one of the underlying columns in the view changes. As a result, updates to the underlying table are not always output. Use `NOFILTERVIEW` to ensure that a record is output whenever the underlying table is updated regardless of whether or not any of the view's columns changed. `FILTERVIEW` ensures that records are output only when one of the columns changed.

Default

```
FILTERVIEW
```

Syntax

```
FILTERVIEW | NOFILTERVIEW
```

FLUSHCHECKPOINT | NOFLUSHCHECKPOINT

Valid for

Replicat

Description

Use `FLUSHCHECKPOINT` to control whether Enscribe files are flushed when Replicat records a checkpoint. Specify `FLUSHCHECKPOINT` to guarantee that when a checkpoint is recorded all data replicated before the checkpoint is stored on disk. If `FLUSHCHECKPOINT` is off, data may remain in file buffers.

Although `FLUSHCHECKPOINT` is safer, it can result in significant performance impact to Replicat processes.

Default

```
NOFLUSHCHECKPOINT
```

Syntax

```
FLUSHCHECKPOINT | NOFLUSHCHECKPOINT
```

FLUSHSECS | FLUSHCSECS

Valid for

Extract

Description

Use these parameters to specify the number of seconds (FLUSHSECS) or centiseconds (FLUSHCSECS) for Extract to buffer records before flushing to the current log trail.

Default

2 seconds

Syntax

```
FLUSHSECS seconds | FLUSHCSECS centiseconds
```

seconds
The maximum number of seconds to buffer records before flushing.

centiseconds
The maximum number of centiseconds to buffer records before flushing.

FORCESTOPDELAY

Valid for

Logger

Description

Use FORCESTOPDELAY with the STOPDELAYCSECS keyword to instruct BASELIB to delay stopping a process for specified time to give Logger time to read messages in its $RECEIVE queue. The delay interval is specified with the STOPDELAYCSECS parameter.

If the process being stopped has messages queued on Logger's $RECEIVE queue and that process is stopped before Logger reads them, the message system will remove those messages from the queue and discard them. If this happens, changes that were done to the database will never get logged to the log trail.

FORCESTOPDELAY applies to a STOP, ABEND, or other operating system instruction that stops a process. It also applies to stops issued from an application, an operator issuing a stop command, or to a process issuing a STOP or ABEND.

If the process that is stopping another process is not bound with BASELIB then this parameter cannot delay the stop.

Default

No delay

Syntax

```
FORCESTOPDELAY STOPDELAYCSECS centiseconds
```

centiseconds
The delay interval in centiseconds.

FORCEUSESYSKEY | NOFORCEUSESYSKEY

Valid for

Replicat

Description

Use `FORCEUSESYSKEY` to force Replicat to use the mapped SYSKEY when executing update and delete operations on entry-sequenced SQL tables. For relative SQL tables, the default is to include the syskey unless other `KEYCOLS` have been specified. When using `FORCEUSESYSKEY`, include `SYSKEY` in a `COLMAP` statement to ensure that the proper row is updated or deleted in the target table. `NOFORCEUSESYSKEY` results in `SYSKEY` being omitted from the `WHERE` clause for entry-sequenced SQL updates and deletes (an alternative path is assumed).

For inserts on relative Enscribe files, `FORCEUSESYSKEY` forces the `SYSKEY` to be -1 (insert at end of file). If `FORCEUSESYSKEY` is used on the target file, the source file SYSKEY cannot be used for update and delete operations since there is no assurance the two keys will match. In this case a unique alternate key or index should be used for updates and deletes.

This parameter has no effect on entry-sequenced or key-sequenced Enscribe files, or on standard key-sequenced or cluster key SQL tables.

Do not use when you are using the `MAP KEYCOLS` option.

Default

```
NOFORCEUSESYSKEY
```

Syntax

```
FORCEUSESYSKEY | NOFORCEUSESYSKEY
```

Examples

Example 1
In this example the `SYSKEY` from the source table is stored in the `SAVE_SYSKEY` column when the record is an insert.

```
GETINSERTS
IGNOREUPDATES
IGNOREDELETES
MAP $DATA1.DAT.SOURCE, TARGET $DATA2.DAT.TARGET,
COLMAP (AMOUNT = AMT,
SAVE_SYSKEY = SYSKEY);
```

Example 2
Then the saved `SYSKEY` from the source identifies the target row when the record is an update or delete.

```
FORCEUSESYSKEY
IGNOREINSERTS
GETUPDATES
GETDELETES
MAP $DATA1.DAT.SOURCE, TARGET $DATA2.DAT.TARGET,
COLMAP (SYSKEY = SAVE_SYSKEY,
AMOUNT = AMT);
NOFORCEUSESYSKEY
```

FORMATASCII | NOFORMATASCII

Valid for

Extract

Description

Use `FORMATASCII` to format subsequent output in external ASCII format. `FORMATASCII` applies to all `FILE` or `TABLE` specifications that follow the `FORMATASCII` entry, and can be turned off with the `NOFORMATASCII` parameter.

Using `FORMATASCII`, you can format output compatible with the majority of popular database load utilities and other tools.

You can specify the parameter with or without options. If you don't include options, records are output as follows:

- An operation type character, `I, D, U, V` (insert, delete, update, compressed update).
- A before or after-image indicator, `B` or `A`.
- The file or table name.
- If the `NAMES` option (the default) is selected: field name, field value, field name, field value...
- If the `NONAMES` option is selected: field value, field value... (`NAMES` format is always used for compressed records).
- Between each of the above items, a field delimiter (which defaults to tab).
- A new line character (line feed).

Before the beginning of the first record output for each transaction, a begin transaction record will appear. This record includes the following information:

- The begin transaction indicator, `B`.
- The timestamp at which the record committed.
- The sequence number of the audit trail in which the commit was found.
- The relative byte address (RBA) of the commit record within the audit trail.
- Delimiters following each of the above fields, followed by a new line character.

After the last record in each transaction is output, a commit record will appear. This record contains `C`, the delimiter and a new line character.

Every record in a transaction, and no other records, are contained between the begin and commit indicators. Each combination of commit timestamp and RBA is unique and increases as records are output.

You can customize the output format with options described in the syntax.

Do not use this format if the data is to be processed by Replicat. Replicat expects the default format.

Syntax

```
NOFORMATACSII | FORMATASCII [, option, ...]
```

option is one of the following.

BCP
Formats the output for compatibility with SQL Server's BCP (Bulk Copy Program) high-speed load utility.
Options that can be used with BCP are:

BCPRECORDLIMITER
The option changes the line delimiter from only line feed ('\n') to line feed and carriage return ('\r\n'). Use when sending data to Windows or UNIX SQL in bulk copy format. NonStop will not receive data in this format.

TIMESTAMP {0 | 3 | 6}
Valid when sending data to Windows or UNIX SQL in bulk copy format. The numeric options set the fractional part of the datetime data type:

- 0 — truncate the fractional portion of the timestamp
- 3 — include three digits in the fractional portion
- 6 — include six digits in the fractional portion

For example, the datetime 2013-12-11 18:26:06:35.123456 would be output as 2013-12-11 18:26:06:35 with the 0 option, 2013-12-11 18:26:06:35.123 with the 3 option, and 2013-12-11 18:26:06:35.123456 with 6.
The default format for the datetime data type is 0, truncate the fractional portion.

> **Note:**
>
> FORMATLOCAL must be used with datetime fractional options 3 or 6 for proper processing by the Collector on a Windows or Linux/UNIX system.

COLHDRS
Outputs the table's column names before the data. COLHDRS takes effect only when extracting directly from the table (rather than the TMF audit trails).

{DATE | TIME | TS}
Specifies one of the following:

- DATE — date (year to day),
- TIME — time (year to second) before record data,
- TS — transaction timestamp (year to fraction).

DELIMITER *delimiter* | COLUMNDELIMITER *delimiter* | RECORDDELIMITER delimiter
Each option sets an alternative delimiter for the column or record.
For example:

```
FORMATASCII DELIMITER 0x09 RECORDDELIMITER 0x0a
FORMATASCII COLUMNDELIMITER ',' RECORDDELIMITER ';'
FORMATASCII DELIMITER '/' RECORDDELIMITER 0x0a
```

Note that the character specified for the field delimiter using either `DELIMITER` or `COLUMNDELIMITER` cannot be the same as the character specified for the record using `RECORDDELIMITER`.
The options for setting the delimiters include the following:

`DELIMITER` *`delimiter`*
An alternative field/column delimiter (the default is tab). Use the word `TAB` to delimit with tabs, otherwise use a single character enclosed in single quotes (for example, '/') or a hexadecimal (for example, 0x0a).

`COLUMNDELIMITER` *`delimiter`*
This is an alias of `DELIMITER`.

`RECORDDELIMITER` *`delimiter`*
Sets the record delimiter . The delimiter can be specified as a single printable character enclosed within single quotes (for example, ';') or a hexadecimal (for example, 0x0a).

`EXTRACOLS` *`number_of_columns`*
Includes placeholders for additional columns at the end of each record. Use this when a target table has more columns than the source.

`FILE`
Includes just the file name portion of the file or table (default is the fully qualified file name).

`NAMES | NONAMES`
Includes or excludes column names from the output. For compressed records, column names are included unless you also specify `PLACEHOLDERS`.

`NOHDRFIELDS` *`header_option`*
Suppresses output of transaction information, the operation type character, the before or after-image indicator, and the file or table name.
You can customize header information by including a *`header_option`* as follows:

- `IND` includes the before or after indicator
- `OP` includes the operation type character
- `WHOLEFILE` includes the fully qualified file name
- `FILE` includes the file name portion of the file or table (as specified above)

`NOQUOTE`
Excludes quotation marks from character-type data. The default is to use quotation marks.

`NOSYSKEY`
Omits the record `SYSKEY` (relative or entry key) from the output, if one exists.

`NOTRANSTMTS`
Excludes transaction information.

NULLISSPACE
Outputs `NULL` fields as empty fields. The default is to output null fields as the word `NULL`.

PLACEHOLDERS
Outputs a placeholder for missing fields or columns. For example, if the second and fourth columns are missing in a four column table, the data would appear similar to:

```
'ABC',,123,,
```

SQLLOADER
Generates a file compatible with the Oracle SQL*Loader high-speed data load utility. `SQLLOADER` produces a fixed-length, ASCII-formatted file. Use this option only when one table's data is written to the Oracle GoldenGate trail (usually in the initial-load, `SOURCEISFILE` case).

Example

The following is a sample table and description. `CUSTNAME` is the primary key.

```
$DATA1.TEST.CUSTOMER:
CUSTNAME    CHAR(10)
LOCATION    CHAR(10)
BALANCE     INTEGER
```

The transaction on this table is:

```
BEGIN WORK;
INSERT INTO CUSTOMER VALUES ("Eric", "San Fran", 550);
UPDATE CUSTOMER SET BALANCE = 100 WHERE CUSTNAME = "Eric";
COMMIT WORK;
```

Entering `FORMATASCII` produces:

```
B,2010-02-17:14:09:46.421335,8,1873474,
I,A,\GGS.$DATA1.TEST.CUSTOMER,CUSTNAME,'Eric',LOCATION, 'San Fran',BALANCE,550,
V,A,\GGS.$DATA1.TEST.CUSTOMER,CUSTNAME,'Eric',BALANCE,100, C,
```

Entering `FORMATASCII, NONAMES, DELIMITER '|', FILE` produces:

```
B|2010-02-17:14:09:46.421335|8|1873474|
I|A|CUSTOMER|'Eric'|'San Fran'|550|
V|A|CUSTOMER|CUSTNAME|'Eric'|BALANCE|100|
C|
```

The last record returns column names because the record is a compressed update.

Entering `FORMATASCII, NOHDRFIELDS, FILE, OP, TS, NONAMES, NOQUOTE` produces:

```
I,CUSTOMER,2010-02-17:14:09:46.421335,Eric,San Fran,550,
V,CUSTOMER,2010-02-17:14:09:46.421335,Eric,,100,
```

The absence of the second field in the update record is indicated by two consecutive commas. Ordering of header fields is predetermined and is not affected by the ordering of options.

The ampersand (&) in the parameter entry continues the parameter to the next line in the parameter file.

FORMATLOCAL

Valid for

Extract

Description

Use `FORMATLOCAL` to determine whether formatting extracted data occurs on the source or the target. When data is output on NonStop, formatting is always local.

When formatting remotely, database definitions need to be exported from NonStop to the remote system. Retrieving this information is automatic when formatting is local, eliminating the requirement to keep remote definitions synchronized with the most current definitions.

Default

Format data remotely (offers far better performance)

Syntax

```
FORMATLOCAL
```

FORMATSQL | NOFORMATSQL

Valid for

Extract

Description

Use `FORMATSQL` to format subsequent output records in external `SQL DML` format. This is the format SQL needed to create the operation, that is, an `INSERT`, `UPDATE` or `DELETE` statement. You can apply this format to both SQL and Enscribe records.

`FORMATSQL` applies to all Oracle GoldenGate files specified below the `FORMATSQL` entry. Turn off `FORMATSQL` with the `NOFORMATSQL` parameter.

The output contains the following transaction-related information before the first record output for each transaction:

- the begin transaction indicator, `B`
- the timestamp at which the record committed
- the sequence number of the audit trail in which the commit was found
- the relative byte address (RBA) of the commit record within the audit trail
- commas following each of the above fields
- a new line character

After the last record in each transaction is output, a commit record will appear. This record contains `C`, the delimiter and a new line character.

Every record in a transaction, and no other records, are contained between the begin and commit indicators. Each combination of commit timestamp and RBA is unique and increases as records are output.

Do not use this format if the data will be processed by Replicat. Replicat expects the default format.

Syntax

```
NOFORMATSQL | FORMATSQL {ORACLE | NONAMES | FILE}
```

ORACLE
Records are formatted for compatibility with Oracle databases. Primarily this means that date and time fields are converted to a format suitable to SQL*Plus (for example, `TO_DATE('2010-05-01','YYYY-MM-DD')`.

NONAMES
Omits column names when all columns are present in insert records to conserve space.

FILE
Outputs only the file portion of the NonStop file name in the SQL statement.

FORMATXML

Valid for

Extract

Description

Use the `FORMATXML` parameter to output data in XML format, instead of the default Oracle GoldenGate canonical format. A `FORMATXML` statement affects all extract files or trails that are defined after it.

> **Note:**
>
> Do not use `FORMATXML` if the data will be processed by the Replicat process. Replicat expects the default canonical format. Do not use `FORMATXML` if `FORMATASCII` or `FORMATSQL` is being used.

Default

None

Syntax

```
FORMATXML [option] [, ...]
```

option is one of the following.

INLINEPROPERTIES | NOINLINEPROPERTIES
Controls whether properties are included within the XML tag or written separately. INLINEPROPERTIES is the default.

ONERECPERTRANS
Causes Extract to create one FORMATXML transaction per record if the SOURCEISFILE parameter is present. The default is to create one FORMATXML transaction per file.

TRANS | NOTRANS
Controls whether transaction boundaries and commit timestamps should be included in the XML output. TRANS is the default.

Example

```
FORMATXML NOINLINEPROPERTIES, NOTRANS
```

FUNCTIONSTACKSIZE

Valid for

Extract, Replicat

Description

Use FUNCTIONSTACKSIZE to control the size of the memory stack that is used for processing Oracle GoldenGate functions. You should not need to use this parameter unless Oracle GoldenGate returns a message indicating that the size of the stack should be increased.

The memory stack holds arguments supplied to and from an Oracle GoldenGate function. When a very large number of functions or arguments are used, the size of the stack may need to be increased.

FUNCTIONSTACKSIZE is a global parameter. It affects all clauses in a parameter file.

The default without FUNCTIONSTACKSIZE is 200 arguments, which optimizes the performance of Oracle GoldenGate and its usage of system memory. Increasing this parameter can adversely affect the performance and use of system memory.

FUNCTIONSTACKSIZE must appear in the parameter file before any parameter clauses are listed.

Default

200

Syntax

```
FUNCTIONSTACKSIZE stack_size
```

stack_size
A value between 50 and 5000 that denotes the number of arguments to allow in a parameter clause.

Example

```
FUNCTIONSTACKSIZE 300
```

GETALTKEYS | IGNOREALTKEYS

Valid for

Extract

Description

Use `GETALTKEYS` or `IGNOREALTKEYS` to tell Extract to produce or not produce file create records for alternate key files after it outputs the file create record for the primary file.

- `GETALTKEYS` causes Extract to create primary and alternate key files.
- `IGNOREALTKEYS` causes Extract to create primary key files.

Default

`GETALTKEYS`

Syntax

```
GETALTKEYS | IGNOREALTKEYS
```

GETAPPLOPS | IGNOREAPPLOPS

Valid for

Extract

Description

Use `GETAPPLOPS` or `IGNOREAPPLOPS` to include or exclude records produced by any program except Replicat in a particular Extract file. `GETAPPLOPS` remains in effect until `IGNOREAPPLOPS` is entered. Once entered, `IGNOREAPPLOPS` is in effect until `GETAPPLOPS` is entered.

`IGNOREAPPLOPS` provides a method for isolating database operations performed by Replicat against audited files from other work.

See "GETREPLICATES | IGNOREREPLICATES" for information on how to use `GETREPLICATES` and `IGNOREREPLICATS` to include or exclude records created by Replicat.

Default

`GETAPPLOPS`

Syntax

```
GETAPPLOPS | IGNOREAPPLOPS
```

GETAUXTRAILS | IGNOREAUXTRAILS

Valid for

Extract

Description

Use IGNOREAUXTRAILS to tell Extract to bypass TMF auxiliary TMF audit trails when capturing database changes. If relevant database changes are recorded in the master audit trail, this can have significant performance advantages.

IGNOREAUXTRAILS can specify particular trails to ignore, rather than all the trails.

Default

GETAUXTRAILS

Syntax

IGNOREAUXTRAILS **aux_trail_num**

aux_trail_num
A value between 0 and 15.

Example

The first example ignores all auxiliary TMF audit trails, while the second example ignores auxiliary trails AUX02 and AUX03.

```
IGNOREAUXTRAILS
IGNOREAUXTRAILS 2, 3
```

GETCOMPS | IGNORECOMPS

Valid for

Extract

Description

Use GETCOMPS or IGNORECOMPS to include or exclude compressed update records for the specified EXTFILE. GETCOMPS remains in effect until IGNORECOMPS is entered. Once entered, IGNORECOMPS remains in effect until GETCOMPS is entered.

Default

GETCOMPS

Syntax

GETCOMPS | IGNORECOMPS

Example

This example includes compressed update records in EXTRACT1 and EXTRACT3, but excludes compressed update records in EXTRACT2.

```
GETCOMPS
EXTFILE $DATA1.EXTDAT.EXTRACT1
        TABLE $DATA2.FINANCE.ACCOUNTS;
IGNORECOMPS
EXTFILE $DATA1.EXTDAT.EXTRACT2
        TABLE $DATA2.FINANCE.ACCOUNTS;
```

```
GETCOMPS
EXTFILE $DATA1.EXTDAT.EXTRACT3
        TABLE $DATA2.FINANCE.ACCOUNTS;
```

GETCREATES | IGNORECREATES

Valid for

Extract, Replicat

Description

Use `GETCREATES` or `IGNORECREATES` to include or exclude file create records from the specified `EXTFILE`. `GETCREATES` remains in effect until `IGNORECREATES` is entered. Once entered, `IGNORECREATES` remains in effect until `GETCREATES` is entered.

Default

```
IGNORECREATES
```

Syntax

```
GETCREATES | IGNORECREATES
```

Example

This example separates inserts, updates and creates into three files, `EXTRACT1`, `EXTRACT2`, and `EXTRACT3`.

```
IGNOREUPDATES
IGNORECREATES
GETINSERTS
EXTFILE $DATA1.EXTDAT.EXTRACT1
TABLE $DATA2.FINANCE.ACCOUNTS;
IGNOREINSERTS
GETUPDATES
IGNORECREATES
EXTFILE $DATA1.EXTDAT.EXTRACT2
TABLE $DATA2.FINANCE.ACCOUNTS;
IGNOREINSERTS
IGNOREUPDATES
GETCREATES
EXTFILE $DATA1.EXTDAT.EXTRACT3
TABLE $DATA2.FINANCE.ACCOUNTS;
```

GETDEFAULTS

Valid for

Extract, Replicat

Description

Use `GETDEFAULTS` to reset Extract parameters to their original default settings. This is useful if you have changed multiple Extract parameters, and now wish to reverse those changes.

Using `GETDEFAULTS` sets the following parameters to `ON`:

```
GETINSERTS
GETUPDATES
GETDELETES
GETUPDATEAFTERS
GETCOMPS
GETPURGEDATAS
GETALTKEYS
```

Using `GETDEFAULTS` sets the following parameters to `OFF`:

```
GETUPDATEBEFORES
GETNETCHANGES
GETPURGES
GETCREATES
GETALTERS
GETRENAMES
```

Syntax

```
GETDEFAULTS
```

GETDELETES | IGNOREDELETES

Valid for

Extract, Replicat

Description

Use `GETDELETES` or `IGNOREDELETES` to include or exclude delete records from the specified data source. `GETDELETES` is in effect until `IGNOREDELETES` is entered. Once entered, `IGNOREDELETES` is in effect until `GETDELETES` is entered.

Default

```
GETDELETES
```

Syntax

```
GETDELETES | IGNOREDELETES
```

Example

This example separates inserts, updates and deletes into three files.

```
IGNOREUPDATES
IGNOREDELETES
GETINSERTS
EXTFILE $DATA1.EXTDAT.EXTRACT1
TABLE $DATA2.FINANCE.ACCOUNTS;
IGNOREINSERTS
GETUPDATES
IGNOREDELETES
EXTFILE $DATA1.EXTDAT.EXTRACT2
TABLE $DATA2.FINANCE.ACCOUNTS;
IGNOREINSERTS
IGNOREUPDATES
GETDELETES
EXTFILE $DATA1.EXTDAT.EXTRACT3
TABLE $DATA2.FINANCE.ACCOUNTS;
```

GETENV

Valid for

Extract, Replicat

Description

Use GETENV to retrieve environment variables that were set with the SETENV parameter. The retrieved results can be used as input parameters for stored procedures or macros. The results are printed to screen and the Extract report file. Use one GETENV statement per variable to be retrieved. See ""GETENV"" for more information on the GETENV variables.

Default

None

Syntax

```
GETENV (environment_variable)
```

environment_variable
The name of the environment variable

Example

```
GETENV TRAIL1
```

GETFILEOPS | IGNOREFILEOPS

Valid for

Extract, Replicat

Description

Use GETFILEOPS or IGNOREFILEOPS to include or exclude file-level operations in the current EXTFILE. IGNOREFILEOPS remains in effect until GETFILEOPS is entered. Once entered, GETFILEOPS remains in effect until IGNOREFILEOPS is entered.

File-level operations that are included or excluded are CREATE, ALTER, PURGE, PURGEDATA, RENAME and some SETMODE, CONTROL and file label change operations. You must specify GETFILEOPS to extract bulk I/O records. When extracting from the TMF audit trail, only CREATE and PURGE records are available. For CREATE operations on audited files, Extract needs the file on disk to extract the record.

To implement GETFILEOPS or IGNOREFILEOPS, specify the desired parameter on one line, followed by the operation to include or exclude as shown in the syntax.

Using GETFILEOPS turns on the following values:

```
GETPURGES
GETPURGEDATAS
GETRENAMES
GETALTERS
```

```
GETCREATES
GETCHANGELABELS
GETSETMODES
GETUNSTRUCTOPS
GETCLOSES
```

Default

```
IGNOREFILEOPS
```

Syntax

```
GETFILEOPS operation | IGNOREFILEOPS operation
```

GETINSERTS | IGNOREINSERTS

Valid for

Extract, Replicat

Description

Use `GETINSERTS` or `IGNOREINSERTS` to include or exclude insert records in the current data source. `GETINSERTS` remains in effect until `IGNOREINSERTS` is entered. Once entered, `IGNOREINSERTS` remains in effect until `GETINSERTS` is entered.

Default

```
GETINSERTS
```

Syntax

```
GETINSERTS | IGNOREINSERTS
```

GETMARKERS | IGNOREMARKERS

Valid for

Extract

Description

Use `GETMARKERS` or `IGNOREMARKERS` to include or exclude marker records from the specified `EXTFILE`.

Marker records are special audit records created by users with the GGSCI command `ADD MARKER`. You can use marker records to identify application-specific critical points in Extract and Replicat processing.

The most common use of a marker is to identify a point at which all data has been replicated from a source to a target database. To do this, an operator brings down application activity against the source database, then adds a marker. After seeing a message that the marker was processed by Replicat, you can safely assume all records from the source database have been processed. At this point, for example, a hot site switch to the backup database could safely occur.

Default

GETMARKERS

Syntax

GETMARKERS | IGNOREMARKERS

GETNETCHANGES | IGNORENETCHANGES

Valid for

Extract

Description

Use GETNETCHANGES to retrieve fields that were changed in a particular update record, plus the primary key. Use IGNORENETCHANGES to output all fields, changed or not.

GETNETCHANGES remains in effect until IGNORENETCHANGES is entered. Once entered, IGNORENETCHANGES remains in effect until GETNETCHANGES is entered.

GETNETCHANGES guarantees that only changed records are retrieved. Compressed update records do not guarantee this will happen, because some update records do not actually change fields and will not be returned (for example, UPDATE TABX SET PRICE = 10 WHERE PRICE = 10). This is common in applications that use, for example, a generic update of each column to accomplish all update tasks, regardless of what is changed.

GETNETCHANGES retrieves and compares both before and after-images. This can result in a system resource usage increase.

Default

IGNORENETCHANGES

Syntax

GETNETCHANGES | IGNORENETCHANGES

GETNETWORKALTFILENAMES | IGNORENETWORKALTFILENAMES

Valid for

Replicat

Description

GETNETWORKALTFILENAMES lets the file system qualify the alternate key file names with the local node name. Use IGNORENETWORKALTFILENAMES to tell the file system not to qualify the alternate key file names with the local node name. You can add this parameter as part of a MAP statement or toggle it around MAP statements.

Default

GETNETWORKALTFILENAMES

Syntax

GETNETWORKALTFILENAMES | IGNORENETWORKALTFILENAMES

GETNEWCOLUMNS | IGNORENEWCOLUMNS

Valid for

Extract, Replicat

Description

Use GETNEWCOLUMNS to retrieve records from the TMF audit trails that describe new SQL column changes. GETNEWCOLUMNS enables two different activities:

- Updates table definitions in Extract and downstream in Replicat, without bringing either process down first (NonStop replication only). This lets you replicate columns added to a table after Oracle GoldenGate startup.
- Downstream in Replicat (NonStop only), replicates the statement that created the column.

Apply GETNEWCOLUMNS to files or Oracle GoldenGate trails individually. This means that you can withhold new column records from specific Extract files. To ensure that the column exists in the target database before replicating into it, it is recommended that GETNEWCOLUMNS apply to all files in the Oracle GoldenGate trail.

Default

GETNEWCOLUMNS

Syntax

GETNEWCOLUMNS | IGNORENEWCOLUMNS

GETPARTONLYPURGEDATAS | IGNOREPARTONLYPURGEDATAS

Valid for

Extract, Replicat

Description

Use GETPARTONLYPURGEDATAS or IGNOREPARTONLYPURGEDATAS to include or exclude PARTONLY PURGEDATA operations. From the point it is entered in the parameter file, the parameter remains in effect until its opposite is encountered. If GETPARTONLYPURGEDATAS is entered, for example, it remains in effect until an IGNOREPARTONLYPURGEDATAS.

GETPARTONLYPURGEDATAS is required to capture PURGEDATA operations on Enhanced Key Sequenced Enscribe files (files that have more than 16 partitions.)

> **Note:**
>
> If a file has more than 16 partitions, PURGEDATA operations can only be captured on a per-partition basis because the primary partition will always be empty.

Before using GETPARTONLYPURGEDATAS carefully consider the following restrictions.

- GETPARTONLYPURGEDATAS applies only to like-to-like replication and cannot be used in heterogeneous configurations.
- GETPARTONLYPURGEDATAS is not supported for bi-directional configurations.
- When you use GETPARTONLYPURGEDATAS, the key range of the partitions on the target must match the key range of those on the source. The PARTONLY PURGEDATA operation will be rejected if the source partition file is out of range for the target file of the same number.
- GETPARTONLYPURGEDATAS is not supported for non-audited files.
- GETPARTONLYPURGEDATAS is not supported for SQL tables.
- If you want an Extract pump to pass on PARTONLY PURGEDATA operations, you must explicitly specify GETPARTONLYPURGEDATAS.

GETPARTONLYPURGEDATAS has no effect on GETPURGEDATAS, and neither GETDEFAULTS nor GETFILEOPS have an effect on GETPARTONLYPURGEDATAS.

The following parameters turn off (disable) GETPARTONLYPURGEDATAS:

- IGNOREFILEOPS
- IGNOREPURGEDATAS

> **Note:**
>
> If PARTONLY PURGEDATA operations are received by older (previous to version 9.5) or open systems Replicats, there will be no attempt to process the PARTONLY PURGEDATA operations.

Default

IGNOREPARTONLYPURGEDATA

Syntax

GETPARTONLYPURGEDATAS | IGNOREPARTONLYPURGEDATAS

GETPURGES | IGNOREPURGES

Valid for

Extract, Replicat

Description

Includes or excludes file purge operations.

Default

IGNOREPURGES

Syntax

GETPURGES | IGNOREPURGES

GETPURGEDATAS | IGNOREPURGEDATAS

Valid for

Extract, Replicat

Description

Use GETPURGEDATAS or IGNOREPURGEDATAS to control whether Extract writes purge data operations to a trail.

Oracle GoldenGate supports PURGEDATA for Enscribe file base tables and complete table PURGEDATA for SQL/MP tables.

GETPURGEDATAS is not supported for audited files in a bidirectional configuration. In this case IGNOREPURGEDATAS must be added to the Extract parameters.

> **Note:**
>
> GETPURGEDATAS captures full file PURGEDATA operations. If the primary partition is empty, however, GETPURGEDATAS will not capture PURGEDATA operations for the secondary partitions. In this case, the PURGEDATA for the secondary partitions will be picked up only by GETPARTONLYPURGEDATAS.

Default

GETPURGEDATAS

Syntax

GETPURGEDATAS | IGNOREPURGEDATAS

GETRENAMES | IGNORERENAMES

Valid for

Extract, Replicat

Description

Includes or excludes file rename records.

Default

IGNORENAMES is the default unless GETFILEOPS is used.

Syntax

```
GETRENAMES | IGNORERENAMES
```

GETREPLICATES | IGNOREREPLICATES

Valid for

Extract

Description

Use GETREPLICATES or IGNOREREPLICATES to include or exclude records created by Replicat. GETREPLICATES remains in effect until IGNOREREPLICATES is entered. Once entered, IGNOREREPLICATES remains in effect until GETREPLICATES is entered.

Use IGNOREREPLICATES when two files are delivering different data to each other. For example, assume A replicates to B and B replicates to A. When A sends an insert record to B, Extract detects the replicated record and extracts it from B. In turn, Replicat replicates it back to A, which produces an error (duplicate record). Specify IGNOREREPLICATES to discard the replicated record.

GETREPLICATES can cascade replication through a chain of files. For example, if A replicates to B, and B replicates to C, IGNOREREPLICATES would prevent data from A being replicated at C. GETREPLICATES enables this to happen.

Even with GETREPLICATES set, you can avoid delivering back to the source by using WHERE or FILTER clauses in Extract or Replicat.

See "GETAPPLOPS | IGNOREAPPLOPS " for information on using GETAPPLOPS and IGNOREAPPLOPS to include or exclude records not created by Replicat.

Default

```
IGNOREREPLICATES
```

Syntax

```
GETREPLICATES | IGNOREREPLICATES
```

GETROLLBACKS | IGNOREROLLBACKS

Valid for

Extract

Description

Use GETROLLBACKS or IGNOREROLLBACKS to include or exclude records created by TMF rollback processes.

Use GETROLLBACKS only when performing change synchronization while extracting initial-load data. If you perform initial load while the source database remains up, GETROLLBACKS guarantees that aborted transactions occurring during the initial extraction phase are rolled back on the target database. This is because initial extraction uses browse access against the source table, which can result in retrieval of records that are later rolled back.

After initial load is complete, issue the SEND EXTRACT *group_name* IGNOREROLLBACKS command from GGSCI.

GETROLLBACKS is only required when the initial-load method uses a bulk method, such as BACKUP, to extract the data or when using Oracle GoldenGate with the FASTUNLOADSHARED parameter.

Default

IGNOREROLLBACKS

Syntax

GETROLLBACKS | IGNOREROLLBACKS

GETUPDATEAFTERS | IGNOREUPDATEAFTERS

Valid for

Extract, Replicat

Description

Use GETUPDATEAFTERS or IGNOREUPDATEAFTERS to include or exclude after-images of update records from a specified file in an Oracle GoldenGate trail. After-images contain record details after an update (as opposed to before-images). GETUPDATEAFTERS remains in effect until IGNOREUPDATEAFTERS is entered. Once entered, IGNOREUPDATEAFTERS remains in effect until GETUPDATEAFTERS is entered.

Default

GETUPDATEAFTERS

Syntax

GETUPDATEAFTERS | IGNOREUPDATEAFTERS

GETUPDATEBEFORES | IGNOREUPDATEBEFORES

Valid for

Extract, Replicat

Description

Use GETUPDATEBEFORES or IGNOREUPDATEBEFORES to include or exclude before-images of update records from the specified EXTFILE. Before-images contain record details before an update (as opposed to after-images). IGNOREUPDATEBEFORES remains in effect until

GETUPDATEBEFORES is entered. Once entered, GETUPDATEBEFORES remains in effect until IGNOREUPDATEBEFORES is entered.

Within your user exit code, you can compare before-images with after-images to identify the net results of a transaction, rather than the final state of a record. For example, if a BALANCE field is $100 before a transaction and $120 after, comparison would show the difference of $20 occurring in that transaction.

Default

IGNOREUPDATEBEFORES

Syntax

GETUPDATEBEFORES | IGNOREUPDATEBEFORES

GETUPDATES | IGNOREUPDATES

Valid for

Extract, Replicat

Description

Use GETUPDATES or IGNOREUPDATES to include or exclude update records from the specified data source. GETUPDATES remains in effect until IGNOREUPDATES is entered. Once entered, IGNOREUPDATES remains in effect until GETUPDATES is entered. If you specify IGNOREUPDATES, compressed updates are ignored as well.

Default

GETUPDATES

Syntax

GETUPDATES | IGNOREUPDATES

GROUPTRANSOPS

Valid for

Replicat

Description

Use GROUPTRANSOPS to group transactions when applying changes to the target database. Small transactions from the source database are grouped into a single large transaction on the target side, which can result in significant performance benefits. In general, GROUPTRANSOPS speeds up processing for smaller transaction sizes.

With GROUPTRANSOPS, the integrity of the original transaction is preserved at the target. GROUPTRANSOPS controls the frequency of checkpoints written to keep track of activity, minimizing I/O (a checkpoint is written and flushed to disk after each transaction completes).

Default

1000

Syntax

```
GROUPTRANSOPS min_op_count
```

min_op_count
The minimum number of operations to be applied in a transaction. Avoid setting *min_op_count* to an arbitrarily high number as tests show that the benefits diminish after about 25-100 operations. Transactions commit before *min_op_count* when appropriate, such as when end-of-file is reached on the trail file.

HANDLECOLLISIONS | NOHANDLECOLLISIONS

Valid for

Replicat

Description

Use HANDLECOLLISIONS to overlay duplicate records into the target database and ignore missing record errors.

If your installation collects data from several sources to update the same record on the same target, data can be lost. Ideally, applications should ensure that each source file manages a specific range of keys. If this is not the case, there can be conflicts on 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 results.

Replicat supplies the HANDLECOLLISIONS parameter to overlay duplicate records with the latest version of the record, even if they key exists. HANDLECOLLISIONS ignores missing update and delete conditions.

HANDLECOLLISIONS is especially useful during the initial load of a target table while the source table remains online. In this phase, the following steps occur:

1. Processing initial data to the target. When the source database remains online, this step will not read everything. Step 1 begins at time 0.
2. Extracting changes that occurred since the beginning of Step 1. Step 2 begins at time 0.
3. After Step 1 completes (time 1), processing of changes extracted in Step 2 (finishes at time 2).

In Step 3, duplicate errors are possible because both Step 1 and Step 2 may extract the same insert records (which occurred since time 0). In such cases, you can use the change record. While you could ignore the duplicate, overlaying the change is safer from an operational standpoint.

There may be instances of missing records when an update or delete operation is attempted on the target table. This is considered normal, since the following chain of events may have occurred:

- Update of record A in source table.
- Delete of record A in source table.

- Extracting update A by Extract process (Step 2).
- Initial-load extraction (Step 1) sees no trace of A, therefore A never inserted into target by initial-load processing.
- Replicating update A attempted in Step 3, but record is missing. Delete of A will result in a missing record error as well.

When all changes have been extracted and applied (time 2), HANDLECOLLLIONS is no longer required. Turn off HANDLECOLLISIONS while Replicat remains online with the GGSCI SEND REPLICAT NOHANDLECOLLISIONS command.

HANDLECOLLISIONS can also be turned off by including NOHANDLECOLLISIONS in the parameter file.

Default

```
NOHANDLECOLLISIONS
```

Syntax

```
HANDLECOLLISIONS | NOHANDLECOLLISIONS
```

Example

The following turns HANDLECOLLISIONS on for TARGET1 and off for TARGET2.

```
HANDLECOLLISIONS
MAP $DATA1.DAT.SOURCE1, TARGET $DATA2.DAT.TARGET1;
NOHANDLECOLLISIONS
MAP $DATA1.DAT.SOURCE2, TARGET $DATA2.DAT.TARGET2;
```

HEARTBEAT

Valid for

GLOBALS, Logger, Extract

Description

Use HEARTBEAT to write a small record to the trail to make sure that Extract or Logger can send data to a remote system. For TMF audited processing, HEARTBEAT can be set in the GLOBALS or the Extract parameter file. For non-audited processing it can be set in the Logger parameter file.

In the Extract or GLOBALS parameter file, HEARTBEAT can be configured to send this small record at the interval you choose, up to 120 seconds. In the Logger file, the record is sent every 60 seconds. The Logger parameter does not allow you to configure HEARTBEAT at a frequency you choose.

Default

Do not issue HEARTBEAT records.

Syntax

For Logger:

```
HEARTBEAT
```

For GLOBALS and Extract:

```
HEARTBEAT seconds
```

seconds
The time interval, in seconds, to write HEARTBEAT records. The value set in the Extract parameter file will override a setting in GLOBALS.

HOMETERM

Valid for

GLOBALS

Description

Use HOMETERM to change the default Oracle GoldenGate system home terminal to another terminal.

Default

The terminal residing on the system on which Oracle GoldenGate is installed.

Syntax

```
HOMETERM home_terminal_name
```

home_terminal_name
Specify a terminal, such as $VHS.

HOMETERMMESSAGES

Valid for

GLOBALS

Description

Use HOMETERMMESSAGES to suppress or direct Oracle GoldenGate messages to the home terminal. By default, Oracle GoldenGate processes write messages to EMS, but not to the home terminal.

This parameter can be overridden in other parameter files by specifying a value for HOMETERMMESSAGES that is different from the GLOBALS value.

Default

NO

Syntax

```
HOMETERMMESSAGES {YES | NO}
```

YES | NO
Specify YES to write to the home terminal and EMS. The default is NO, suppress writing to the home terminal and write only to EMS.

HOST

Valid for

GLOBALS

Description

The HOST parameter does the following:

- Identifies the remote NonStop system to communicate with GGSCI.
- Helps Oracle GoldenGate resolve file names when delivering data over TCP/IP. For installations without Expand connections, the HOST parameter is required for mapping node numbers to node names. If an Expand link exists between systems, use the NODENUM parameter instead of HOST.
- Identifies the NonStop volume and subvolume where the Oracle GoldenGate environment can be found for remote nodes. When the NETWORKCHECKPOINTS parameter is set, Replicat uses this information to identify the local checkpoint file to update for replication to a file partition. This then allows Replicat to recognize replicated data in a bi-directional system.

Syntax

```
HOST system_name
[, GGSSUBVOL subvolume]
[, NODENUM node_number]
```

system_name
Identifies the host system name.

GGSSUBVOL ***subvolume***
Identifies the subvolume. Required with NODENUM.

NODENUM ***node_number***
Identifies the NonStop node.

Examples

Example 1
The first example below identifies an Oracle GoldenGate installation on remote node \SF as subvolume $DATA3.GGS. The second identifies node number 109 as NonStop system \NY.

```
HOST \SF, GGSSUBVOL $DATA3.GGS
HOST \NY, NODENUM 109
```

Example 2
This example identifies multiple nodes that contain data and an Oracle GoldenGate installation.

```
HOST \SF, NODENUM 112, GGSSUBVOL $DATA3.GGS
HOST \BACK, NODENUM 113, GGSSUBVOL $VOL03.GGS
HOST \LA, NODENUM 114, GGSSUBVOL $DATA6.GGS
HOST \NY, NODENUM 115, GGSSUBVOL $PROD1.GGS
```

IGNOREPARAMERROR

Valid for

GLOBALS

Description

Use IGNOREPARAMERROR to prevent programs from terminating abnormally (abending) when an unrecognized or bad GLOBALS entry is encountered.

This can be useful when different versions of Oracle GoldenGate modules are used in the same installation, and a new GLOBALS parameter has been added since the older module was created.

Default

NO

Syntax

```
IGNOREPARAMERROR {YES | NO}
```

IGNORETMFDUMPS

Valid for

GLOBALS

Description

Use IGNORETMFDUMPS to prevent GGSCI and the Extract and Replicat processes from capturing information about TMF disk and tape dumps. Specifying IGNORETMFDUMPS YES avoids the time-consuming process of updating internal tables containing TMF catalog information. Updating catalog information can also cause SPI error 291 on systems that are processing near their limit.

Default

NO

Syntax

```
IGNORETMFDUMPS {YES | NO}
```

INCLUDE

Valid for

Extract, Replicat

Description

Use INCLUDE to include a macro library in a parameter file.

Default

None (no file included)

Syntax

```
INCLUDE file_name
```

file_name
The full path to the library file.

Example

The following example includes macro library `DATELIB`.

```
INCLUDE $DATA4.GGSMACR.DATELIB
```

INCLUDESOURCEAPPINFO | EXCLUDESOURCEAPPINFO

Valid for

Logger

Description

Use `INCLUDESOURCEAPPINFO` to capture the name and process ID (PID) of the source application program that alters the files being logged for non-audited Enscribe. The program name and process name are added to the trail as token elements `PROGRAMNAME` and `PROCESSNAME`.

The information can be retrieved by using `@GETENV`. For example `@GETENV ("RECORD", "PROCESSNAME")` retrieves the name of the process.

Default

```
EXCLUDESOURCEAPPINFO
```

Syntax

```
INCLUDESOURCEAPPINFO | EXCLUDESOURCEAPPINFO
```

INSERTALLRECORDS | NOINSERTALLRECORDS

Valid for

Replicat

Description

Use `INSERTALLRECORDS` to apply all record types as inserts in the target. Normally, Replicat applies inserts, updates and deletes to the target database as they occur on the original database. Consider the following sequence of transactions:

Sequence	Operation	Table	ID	BALANCE
1	INSERT	CUSTOMER	DAVE	1000
2	UPDATE	CUSTOMER	DAVE	900
3	UPDATE	CUSTOMER	DAVE	1250
4	DELETE	CUSTOMER	DAVE	1250

These operations, after replication, would leave no trace of the ID `DAVE`. No transaction information would be kept, only the ending balance. Therefore, we would have no knowledge that the first update reduced `BALANCE` by 100, or that the second update increased `BALANCE` by 350. Finally, we would have no idea that `DAVE` was ever deleted from the database, or what his ending `BALANCE` was.

`INSERTALLRECORDS` allows this information to be recorded. Instead of applying updates and deletes as they originally occurred, `INSERTALLRECORDS` forces Replicat to insert the information as a new record into the target table. `INSERTALLRECORDS` results in the storage of all images—before and after—into the target database.

Combining this information with special transaction information provides a way to create a database that contains more useful information. You can add special column values related to each transaction to the target data to make better reporting possible.

Using `INSERTALLRECORDS` increases the size of your target tables, so you should only enable it where complete records are required. `INSERTALLRECORDS` applies to all tables listed below it in the parameter file, until you turn it off again by specifying `NOINSERTALLRECORDS`. `INSERTALLRECORDS` can also be limited to a specific file or table by using it as an option under the `MAP` parameter.

Default

```
NOINSERTALLRECORDS
```

Syntax

```
INSERTALLRECORDS | NOINSERTALLRECORDS
```

Example

Example 1
To build a more transaction-oriented view of customers, rather than the latest state of the database, enter the following into the parameter file:

```
INSERTALLRECORDS
MAP =CUSTOMER, TARGET =CUSTHIST,
COLMAP (USEDEFAULTS,
TS = @GETENV ("GGHEADER", "COMMITTIMESTAMP"),
BEF_AFT = @GETENV ("GGHEADER", "BEFOREAFTERINDICATOR"),
OP_TYPE = @GETENV ("GGHEADER", "OPTYPE"),
ID = ID,
BALANCE = BALANCE);
```

This generates the net effect of each transaction, as in the following SQL query that returns the net sum of each transaction along with the time of the transaction and the customer ID.

```
SELECT A.ID, A.TS, A.BALANCE - B.BALANCE
FROM CUSTHIST A, CUSTHIST B
WHERE A.ID = B.ID AND A.TS = B.TS AND
A.OP_TYPE = 'A' AND B.OP_TYPE = 'B';
```

Example 2
The following example applies all record types as inserts only for the `$DATA3.TARGET.HISTORD` order history file.

```
MAP $DATA.SOURCE.CUSTORD TARGET $DATA3.TARGET.HISTORD
INSERTALLRECORDS
COLMAP (USEDEFAULTS,
TRAN_TIME = @GETENV ("GGHEADER", "COMMITTIMESTAMP"),
BEF_AFT = @GETENV ("GGHEADER", "BEFOREAFTERINDICATOR"),
OP_TYPE = @GETENV ("GGHEADER", "OPTYPE"));
```

INSERTDELETES | NOINSERTDELETES

Valid for

Replicat

Description

Use `INSERTDELETES` to convert all delete records to insert operations. `INSERTDELETES` applies to all maps specified below it in the parameter file until `NOINSERTDELETES` is specified.

Default

```
NOINSERTDELETES
```

Syntax

```
NSERTDELETES | NOINSERTDELETES
```

INSERTMISSINGUPDATES | NOINSERTMISSINGUPDATES

Valid for

Replicat

Description

Use `INSERTMISSINGUPDATES` to insert a record when Replicat attempts to apply an update to a record that is missing from the target. This applies only to uncompressed updates, including full images fetched by Extract using `FETCHCOMPS`.

When `NOINSERTMISSINGUPDATES` is in effect, this situation causes a missing record error and the transaction may abort depending on `REPERROR` settings. `INSERTMISSINGUPDATES`

applies to all maps specified below it in the parameter file until NOINSERTMISSINGUPDATES is specified.

Default

NOINSERTMISSINGUPDATES

Syntax

INSERTMISSINGUPDATES | NOINSERTMISSINGDATES

INSERTUPDATES | NOINSERTUPDATES

Valid for

Replicat

Description

Use INSERTUPDATES to convert uncompressed update records to insert operations. INSERTUPDATES applies to all maps specified below it in the parameter file until NOINSERTUPDATES is specified.

Default

NOINSERTUPDATES

Syntax

INSERTUPDATES | NOINSERTUPDATES

IPINTERFACE

Valid for

Manager, Extract

Description

Under NonStop parallel TCP/IP architecture, a process can have multiple interfaces available. Use IPINTERFACE to restrict the Manager or Extract process to the interface specified by an input IP address or DNS name.

If the specified address is not associated with the TCP/IP process, Extract will ABEND with an error and STATUS MANAGER will return an error that varies depending on the address protocol that is used and installed (IPv4 or IPv6). The errors returned by STATUS MANAGER will be something like:

```
Manager process $EXMGR is running (IP socket not open error 4115 (Can't assign
requested address, completing bind)).

Manager process $EXMGR is running(Process $ZTC0, IP (null) port 12345).

Manager process $EXMGR is running(IP socket not open Invalid function argument).
```

Default

Handles the requesting IP address.

Syntax

```
IPINTERFACE {ip_address | dns_name}
```

ip_address
The IP address to which Manager or Extract is to be restricted.

dns_name
The domain name to which Manager or Extract is to be restricted.

Examples

The IP address can be restricted either by using `IPINTERFACE` or by using the optional *`@ip_address`* with `TCPIPPROCESSNAME` as shown in the examples below.

Example 1
The IP address can be specified by using the `IPINTERFACE` parameter as shown in the example below.

```
TCPIPPROCESSNAME $ZTC1
IPINTERFACE 192.0.2.2
```

Example 2
Or the IP address can be specified by attaching the IP address to the `TCPIPPROCESSNAME` parameter as shown in the example below.

```
TCPIPPROCESSNAME $ZTC1@192.0.2.2
```

LAGCRITICAL

Valid for

Manager

Description

Use `LAGCRITICALCSECS`, `LAGCRITICALSECONDS`, `LAGCRITICALMINUTES`, or `LAGCRITICALHOURS` to specify the time interval at which Extract or Replicat processing lag is reported to the event log as a critical message. For example, `LAGCRITICALMINUTES 5` specifies a reporting interval of five minutes. Likewise, `LAGCRITICALSECONDS` 20 specifies a reporting interval of 20 seconds; `LAGCRITICALHOURS 2` specifes two hours.

When using a `LAGCRITICAL` parameter, specify `LAGREPORT` to determine how often the lag times are evaluated and reported.

Syntax

```
LAGCRITICALCSECS csecs [,process_type group_name] [,IGNORETASKS|REPORTTASKS]
[,IGNORESTOPPED|REPORTSTOPPED]|
LAGCRITICALSECONDS seconds [,process_type group_name] [,IGNORETASKS|REPORTTASKS]
[,IGNORESTOPPED|REPORTSTOPPED]|
LAGCRITICALMINUTES minutes [,process_type group_name] [,IGNORETASKS|REPORTTASKS]
[,IGNORESTOPPED|REPORTSTOPPED]|
```

```
LAGCRITICALHOURS hours [,process_type group_name] [,IGNORETASKS|REPORTTASKS]
[,IGNORESTOPPED|REPORTSTOPPED]
```

csecs
The reporting interval in centiseconds, as in `LAGCRITICALCSECS` 50.

seconds
The reporting interval, in seconds, as in `LAGCRITICALSECONDS` 10.

minutes
The reporting interval, in minutes, as in `LAGCRITICALMINUTES` 5.

hours
The reporting interval, in hours, as in `LAGCRITICALHOURS` 1.

process_type group_name
Specifies the process for which the lag will be reported.

process_type
Must be `EXTRACT` or `REPLICAT` or `ER`.

group_name
The name of the Extract group or Replicat group or wildcard group.

IGNORETASKS|REPORTTASKS
By default the task groups are not reported.
By adding the `REPORTTASKS` option, the lag information will be reported for `SOURCEISFILE` and `SPECIALRUN` groups, toggle off with `IGNORETASKS`.

IGNORESTOPPED|REPORTSTOPPED
By default no information is printed for a group in a `STOPPED` state. By adding the `REPORTSTOPPED` option, lag information will be reported, to toggle off with `IGNORESTOPPED`.

LAGINFO

Valid for

Manager

Description

Use `LAGINFOSECONDS`, `LAGINFOMINUTES`, or `LAGINFOHOURS` to determine the point at which lag should be reported to the event log as an informational message. For example, `LAGINFOSECONDS 5` specifies a reporting interval of five seconds. Likewise, `LAGINFOMINUTES 10` specifies a reporting interval of ten minutes; `LAGINFOHOURS 2`, specifies two hours.

When using a `LAGINFO` parameter, specify `LAGREPORT` to determine how often the lag times are evaluated and reported.

Syntax

```
LAGINFOCSECS csecs [,process_type group_name] [,IGNORETASKS|REPORTTASKS]
[,IGNORESTOPPED|REPORTSTOPPED]|
LAGINFOSECONDS seconds [,process_type group_name] [,IGNORETASKS|REPORTTASKS]
[,IGNORESTOPPED|REPORTSTOPPED]|
LAGINFOMINUTES minutes [,process_type group_name] [,IGNORETASKS|REPORTTASKS]
```

```
[,IGNORESTOPPED|REPORTSTOPPED]|
LAGINFOHOURS hours [,process_type group_name] [,IGNORETASKS|REPORTTASKS]
[,IGNORESTOPPED|REPORTSTOPPED]
```

csecs
The reporting interval in centiseconds.

seconds
The reporting interval, in seconds.

minutes
The reporting interval, in minutes.

hours
The reporting interval, in hours.

process_type group_name
Specifies the process for which the lag will be reported.

> ***process_type***
> Must be `EXTRACT` or `REPLICAT` or `ER`.
>
> ***group_name***
> The name of the Extract group or Replicat group or wildcard group.

IGNORETASKS|REPORTTASKS
By default the task groups are not reported.
By adding the `REPORTTASKS` option, the lag information will be reported for `SOURCEISFILE` and `SPECIALRUN` groups, toggle off with `IGNORETASKS`.

IGNORESTOPPED|REPORTSTOPPED
By default no information is printed for a group in a `STOPPED` state. By adding the `REPORTSTOPPED` option, lag information will be reported, to toggle off with `IGNORESTOPPED`.

LAGREPORT

Valid for

Manager

Description

`LAGREPORTMINUTES` and `LAGREPORTHOURS` determine the interval at which Manager checks lag for Extract and Replicat processing.

For example, `LAGREPORTHOURS 1` reports lag every hour. If the lag is greater than the time specified by `LAGCRITICAL`, Manager reports the LAG as critical, otherwise, it reports the lag as an informational message. LAG is not reported if the values are below the threshold set with the `LAGINFO` parameter.

If LAG is being reported from Extract or Replicat, that it is "*Running but has not checkpointed*" in the specified time using either `LAGCRITICAL` or `LAGINFO`, you should send the process a `STATUS` command from GGSCI, and take the appropriate action depending on the reply. Do not stop the process.

Examples:

- A Reply of `Process is suspended, waiting for Resume command` is the result of an `EVENTACTION` triggering a `SUSPEND` and waiting for a `GGSCI RESUME`.
- A reply of `Retrying open error 11 on file` is waiting for someone to create the missing target file for a Replicat to continue.
- A reply of status: `Waiting for more audit` or `Records so far in current transaction` is almost certainly working correctly; possibly waiting on a large transaction.If the problem persists and you are not sure how to resolve it, contact Oracle GoldenGate support.

Syntax

```
LAGREPORTMINUTES minutes | LAGREPORTHOURS hours
```

minutes
The reporting interval, in minutes, as in: `LAGREPORTMINUTES 5`.

hours
The reporting interval, in hours, as in: `LAGREPORTHOURS 1`.

LAGSTATS

Valid for

Extract, Replicat

Description

Use `LAGSTATS` to periodically collect, and optionally report, lag and other performance-related statistics to the report file.

Syntax

```
LAGSTATS
[, INTERVALunit num_units]
[, THRESHOLDunit num_units]
[, REPORT]
```

`INTERVAL` *`unit num_units`*
The interval for which data is collected and optionally reported. For example, `INTERVALSECONDS 30` collects statistics for the last thirty seconds; statistics related to peaks and averages are reset every thirty seconds.
`unit` can be one of:

- `MSECS` (milliseconds)
- `CSECS` (centiseconds)
- `SECONDS`
- `MINUTES`
- `HOURS`
- `DAYS`

THRESHOLD *unit num_units*
Reports percentages of records processed above or below a time lag threshold. For example, THRESHOLDMSECS 500 reports the percentage of records replicated on the target database within 500 milliseconds of entry into the source database. You can specify up to four thresholds.

REPORT
Statistics are output to the report file whenever a reporting interval is passed. If REPORT is not specified, you can retrieve statistics with the GGSCI SEND command, using the LAGSNAPSHOT option.

LIMITRECS

Valid for

Extract

Description

Use LIMITRECS to set a maximum number of records to be extracted from a source table. This parameter has no effect unless you specify SOURCEISFILE. Use LIMITRECS to generate a sample of test data.

Syntax

```
LIMITRECS num_recs
```

num_recs
The number of records to extract.

LIST | NOLIST

Valid for

Extract, Replicat

Description

Use LIST and NOLIST to control whether the macros of a macro library are listed in the report file. Listing can be turned on and off by placing the LIST or NOLIST parameters within the parameter file or within the macro library file. Using NOLIST reduces the size of the report file.

Default

```
LIST
```

Syntax

```
LIST | NOLIST
```

Example

In the following example, NOLIST excludes the macros in the macro library from being listed in the report. Using LIST after the INCLUDE statement restores normal listing for subsequent macros.

```
NOLIST
INCLUDE macro_library
LIST
INCLUDE macro_library
```

LOG

Valid for

Logger

Description

Use `LOG` to identify both a Logger process and the volume to which the Logger process writes data (a *log trail*). The trails hold data for Replicat and are maintained by Oracle GoldenGate management processes.

The number of `LOG` entries in the Logger parameter file indicates the number of Logger processes in the system and the number of log trails. Each Logger process writes data to exactly one log trail. By default, `LOG` process names have the format `$GGLnn`. Therefore, if two `LOG` entries are made in the parameter file, processes `$GGL00` and `$GGL01` perform logging for the system.

The first `LOG` entry must precede all other parameter entries. `LOG` space is allocated after the configuration is processed.

Syntax

```
LOG logtrail_name
[, EXTENTS (primary, secondary, maximum) |  MEGABYTES megabytes]
[, NUMFILES number]
[, NEWFORMAT]
[, OWNER group_number, user_number]
[, PROCESS process_name]
[, RECSIZE bytes| BLOCKSIZE bytes]
[, SECURE "rwep"]
```

logtrail_name
The location of the log trail files for the current Logger process and the two-character process prefix. You can use the Oracle GoldenGate defaults, or specify a custom subvolume or process prefix.
To specify a custom subvolume or a custom process prefix, enter `$vol.subvol.xx`, where `$vol.` is the volume where Oracle GoldenGate is installed, `subvol` is the custom name of your subvolume, and `xx` is the two-character process prefix.
See "Specifying a custom subvolume, process prefix, or process name" for more information.

EXTENTS *primary*, *secondary*, *maximum*
The storage dimensions of each log trail file, where *primary*, *secondary*, *maximum* represents the extents. You must include `EXTENTS` if you omit `MEGABYTES`.

MEGABYTES *megabytes*
The storage in megabytes allocated per log trail file where *megabytes* represents the number of megabytes. You must include `MEGABYTES` if you omit `EXTENTS`.

NUMFILES *number*
The number of files to include in the log trail.

NEWFORMAT
NEWFORMAT produces Oracle GoldenGate version 7 format trails.

OWNER *group_number*, *user_number*
The Guardian group and user that owns the log trail files. The default is the group user running GGSCI.

PROCESS *process_name*
Lets you override the default process name, for example $GGL01. This can be useful when you add or alter an Extract or Replicat group.
For more information, see "Specifying a custom subvolume, process prefix, or process name".

RECSIZE *bytes* | BLOCKSIZE *bytes*
The record or block size in 4096 byte increments up to 56K (57344). It will be rounded to the next higher multiple if the entered value is not a multiple of 4096.

SECURE "*rwep*"
The Guardian security applied to each of the files. Defaults to the default security of the owner.

Examples

Example 1
The following example creates and pre-allocates 10 files sized at 50 megabytes each in $DATA.GGSLOG, owned by SUPER.SUPER with security NUUU, with a prefix of LT.

```
LOG $DATA.GGSLOG.LT, MEGABYTES 50, NUMFILES 10, OWNER 255,255, SECURE "NUUU"
```

Example 2
The following changes the default process name.

```
LOG $DATA.GGSLOG.LT MEGABYTES 500, PROCESS $GGL01, NUMFILES 10, SECURE "NNNN"
```

Specifying a custom subvolume, process prefix, or process name

By default, Oracle GoldenGate defaults to the subvolume GLOGGGL, and assigns a default process prefix and name. For example, files are created in *volume*.GLOGGGL and the default processes would be $GGL00 writing to log trail files AA000000, AA000001 and so on and $GGL01 writing to BB000001, BB000001, etc. To use the Oracle GoldenGate default, enter $vol.GLOGGGL, where $vol is the volume where Oracle GoldenGate is installed.

However, should you change or add a Logger, Extract or Replicat component, you may need to change the subvolume, process prefix, or process name to ensure the correct trails are read.

- To change the subvolume or process prefix, see the *logtrail_name* argument.
- To change the process name, see the PROCESS argument.

LOGFILESBEHIND | LOGFILESBEHINDINFO

Valid for

GLOBALS, Manager

Description

Use LOGFILESBEHIND or LOGFILESBEHINDINFO to report Extract and Replicat processing lags.

Manager can monitor log trails to help prevent premature recycling. Premature recycling can happen when an Extract or Replicat process is down for an extended period and unable to process data written by Logger. Whenever Logger moves to a new file in the log trail sequence, the oldest log file is recycled and the data is lost.

- LOGFILESBEHIND sends a critical message when Extract or Replicat processing lags a specified number of files behind the current log trail file.
- LOGFILESBEHINDINFO sends an informational message when Extract or Replicat falls the specified number of files behind the current log trail file. You can specify both LOGFILESBEHIND and LOGFILESBEHINDINFO.

The messages are generated on the host system of the process.

Default

No report when falling behind

Syntax

```
LOGFILESBEHIND num_files | LOGFILESBEHINDINFO num_files
```

num_files
The number of files falling behind.

Example

For these examples, assume that these parameters are executed on \LA, and also on \LA several Replicat programs deliver data logged to \NY.$DATA1.GLOGGGL.AA.

The log process on \NY is currently logging to file AA000100. If any Replicat at \LA is processing log trail AA000097 or earlier, Manager generates an informational message at \LA. If any Replicat is processing log trail AA000092 or earlier, Manager sends a critical message to the \LA EMS console.

```
LOGFILESBEHINDINFO 3
LOGFILESBEHIND 8
```

LOGFILEOPENS

Valid for

Logger, GLOBALS

Description

Use LOGFILEOPENS to determine how many opens Logger keeps on the current log files.

Default

8

> **Note:**
>
> The default setting is recommended unless otherwise specified by Oracle GoldenGate Support.

Syntax

```
LOGFILEOPENS num_opens
```

num_opens
The number of opens. The maximum is 16.

LOGGERFILENUM

Valid for

Logger

Description

Use `LOGGERFILENUM` when you are running applications with preset file numbers.

As `BASELIB` opens Logger, Logger file numbers can collide with the preset file numbers the application expects to use. `BASELIB` deals with this by opening Logger multiple times until it gets a file number greater than those used. Then `BASELIB` closes the temporary opens containing file numbers that could not be used.

`LOGGERFILENUM` instructs `BASELIB` to force its open on Logger to be a file number greater than the value set for `LOGGERFILENUM`. For example, if the application uses preset file numbers 1 through 10 you would set `LOGGERFILENUM` to a number greater than 10.

This parameter is unnecessary for most installations. It is only required if the user application opens files with a specified file number.

Syntax

```
LOGGERFILENUM file_number
```

file_number
The file number on which BASELIB bases its Logger file numbering sequence.

Example

```
LOGGERFILENUM 30
```

LOGGERFLUSHRECS

Valid for

Logger, `GLOBALS`

Description

Use `LOGGERFLUSHRECS` to specify the number of records for Logger to buffer before flushing to the current log trail. Logger flushes data when the `LOGGERFLUSHRECS` threshold is met.

Default

8

Syntax

```
LOGGERFLUSHRECS num_recs
```

num_recs
The maximum number of records to buffer before a flush occurs.

LOGGERFLUSHSECS | LOGGERFLUSHCSECS

Valid for

Logger, `GLOBALS`

Description

Use `LOGGERFLUSHSECS` or `LOGGERFLUSHCSECS` to specify the number of seconds or centiseconds for Logger to buffer records before flushing to the current log trail.

Default

0.01 second (1 centisecond)

Syntax

```
LOGGERFLUSHSECS seconds | LOGGERFLUSHCSECS centiseconds
```

seconds
The maximum number of seconds to buffer records before flushing.

centiseconds
The maximum number of centiseconds to buffer records before flushing.

LOGGERTIMEOUTSECS

Valid for

Logger

Description

Use `LOGGERTIMEOUTSECS` to specify how long `GGSLIB` waits for a response from Logger before allowing the application to resume normal operations. For example, `LOGGERTIMEOUTSECS 50` specifies that `GGSLIB` waits up to 50 seconds.

After a timeout, GGSLIB stops logging data until either an ALTER LOGGER or START LOGGER is issued from GGSCI. The LOGGERTIMEOUTSECS parameter is global and applies to all Logger processes.

Default

60

Syntax

```
LOGGERTIMEOUTSECS seconds
```

seconds
The number of seconds for GGSLIB to wait for a response from Logger.

LOGGGSCICOMMANDS

Valid for

GLOBALS

Description

Use LOGGGSCICOMMANDS to include or omit user commands to the LOGGGS file. You can view this file with the GGSCI command VIEW GGSEVT.

Default

YES

Syntax

```
LOGGGSCICOMMANDS {YES | NO}
```

LOGON

Valid for

Extract, Replicat

Description

Use LOGON to run Extract or Replicat under an ID different from the process that starts it (normally Manager). Extract and Replicat normally inherit the user ID of the Manager process. This parameter provides password encryption options.

You must place the LOGON parameter near the top of your parameter file so that all other files are created with the correct ownership. For example, a DISCARDFILE may be created with the incorrect user ID and password, causing problems when Extract and/or Replicat try to start up.

Either place the entire LOGON parameter statement on one line, or use an ampersand (&) continuation to split the statement into two or more lines.

If Manager is running under SUPER.SUPER authority, only the *user_id* portion of this parameter is necessary (the comma is still required). If *password* is specified, it is *not* echoed in the report file.

Syntax

```
LOGON user_id, PASSWORD password
[ENCRYPTKEY DEFAULT | ENCRYPTKEY keyname]
```

user_id
The user ID for running Extract.

password
The password for running Extract.

ENCRYPTKEY DEFAULT
Required if the password was encrypted with a default Oracle GoldenGate key by means of the `ENCRYPT PASSWORD ENCRYPTKEY` command without arguments.

ENCRYPTKEY *keyname*
Required if the password was encrypted with a user-defined key by means of the `ENCRYPT PASSWORD ENCRYPTKEY keyname` command. For *keyname*, use the logical name as shown in the `ENCKEYS` file.

Example

Example 1

```
LOGON "super.super", PASSWORD "ggs123"
```

Example 2

```
LOGON super.super, PASSWORD & AACAAAAAAAAAAAIALCKDZIRHOJBHOJUH, ENCRYPTKEY superx128
```

Example 3

```
LOGON super.super, PASSWORD & AACAAAAAAAAAAAIALCKDZIRHOJBHOJUH, ENCRYPTKEY default
```

MACRO

Valid for

Extract, Replicat

Description

Use `MACRO` to create a Oracle GoldenGate macro. For instructions on creating and using Oracle GoldenGate macros, see Using Oracle GoldenGate Macros.

The following syntax *must* be in the order shown and terminated with a semicolon (;).

Syntax

```
MACRO #macro_name
PARAMS (param1, param2 ,...)
BEGIN
macro_body
END;
```

macro_name
The name for the macro. Macro and parameter names are not case-sensitive.

Macro and parameter names must begin with a macro character. The default is the pound (#) character, as in `#macro1` and `#param1`. Anything in the parameter file that begins with the macro character is assumed to be either a macro or a macro parameter.
You can change the macro character with the `MACROCHAR` parameter. Valid macro and parameter characters are alphanumeric and can include the underscore character (_). Parameter or macro names within quotation marks are treated as text and ignored.

`PARAMS (param1, param2, ...)`
Optional. Describes parameters to the macro. Parameter names are not case-sensitive.
Every parameter used in a macro must be declared in the `PARAMS` statement, and when the macro is invoked, the invocation must include a value for each parameter. By default, Oracle GoldenGate permits up to 30 parameters in a `PARAMS` clause, but you can change the default with the `FUNCTIONSTACKSIZE` parameter.

`BEGIN`
Begins the macro body. Must be specified before the macro body.

`macro_body`
Represents the macro body, which consists of one or more statements to be used as parameter file input. The macro body can include simple parameter statements such the first example below, and it can include complex statements like the second example. It also can include invocations of other macros, as in the third example.
Example 1: `COL1 = COL2`
Example 2: `COL1 = #val2`
Example 3: `#colmap(COL1, #sourcecol)`

`END`
Ends the macro definition.

`;`
(Semicolon) Marks the end of the macro parameter.

Examples

Example 1
The following example defines a macro that takes 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;
```

Example 2
The following example defines a macro that does not require parameters.

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

Example 3
The following example defines a macro that calls other macros.

```
MACRO #assign_date
PARAMS (#target_col,#year,#month,#day)
BEGIN
#target_col = #make_date (#year, #month, #day)
END;
```

MACROCHAR

Valid for

Extract, Replicat

Description

Use `MACROCHAR` to change the macro character to something other than #. For example, you might want to change the character when table names include the # character. Anything in the parameter file that begins with the specified macro character is assumed to be either a macro or a macro parameter. The `MACROCHAR` can only be specified once.

To define a different macro character, precede the first `MACRO` statement with the `MACROCHAR` parameter in the parameter file.

Default

(pound symbol)

Syntax

```
MACROCHAR character
```

character
The character to be used as the macro character. Must precede the first macro statement.

Example

In the following example, $ is defined as the macro character.

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

MANAGERREQUIRED

Valid for

GLOBALS

Description

Use MANAGERREQUIRED to specify whether Extract or Replicat can run without a Manager process.

Default

NO

Syntax

```
MANAGERREQUIRED {YES | NO}
```

MAP

Valid for

Replicat

Description

Use MAP to deliver records from a source to the target. Normally, the source is an Oracle GoldenGate trail containing records that were processed by Extract. The MAP parameter is similar to Extract's FILE and TABLE parameters in its mapping capabilities and functionality for executing user exits and stored procedures.

At least one MAP statement is required.

You can invoke MAP more than once in a parameter file, and you can invoke the same MAP argument more than once. This is useful, for example, to split records into different trails to be replicated to different targets.

Syntax

```
MAP source_file_name, TARGET target_file_name
[, EXCEPTIONSONLY]
[, DEF source_ddl_definition]
[, DICTIONARY source_ddl_dictionary]
[, EXITPARAM "exitparam_string"]
[, WHERE (where_condition)]
[, FILTER (expression)]
[, RANGE (range_specification)]
[, COLMAP ([USEDEFAULTS], column_map_specification)]
[, COMPENSCRIBEMAPS | NOCOMPENSCRIBEMAPS]
[, INSERTALLRECORDS]
[, MAPID]
[, PARTIALCOLSOK | NOPARTIALCOLSOK]
[, SQLNAME]
[, USESOURCERECLENGTH]
[, TARGETDEF target_ddl_definition]
[, TARGETDICT target_ddl_dictionary]
[, KEYCOLS (column_list)]
[, USEALTKEY (key_specifier)]
[, UNMAPPEDALTFILECREATES {ALTFILEVOL | PRIMARYVOL}]
[, CREATETEMPLATE file_name]
[, ALTFILECHAR num_chars]
[, GETNETWORKALTFILENAMES | IGNORENETWORKALTFILENAMES]
[, HANDLECOLLISIONS | NOHANDLECOLLISIONS]
```

```
[, DETECTLOCKS]
[, REPERROR (error_number, response)]
[, PARTMAP (source_partition_spec, target_partition_spec)]
[, MAPEXCEPTION (TARGET exception_name, mapping_arguments)]
[, SHOWSYNTAX]
[, SQLEXEC (sqlexec_clause)]
[, EVENTACTIONS (action_options)

;
```

source_file_name
The origin of the record to deliver. Most often, this is the name of the file, table or SQL view from which the record was originally extracted. If Extract performed column mapping on the associated records, however, a different file identifier is attached to reflect the new column structure.

source_file_name can also be an existing define name of `CLASS MAP`, or a wildcard specification.

TARGET *target_file_name*
The name of the target file or table.

DEF *source_ddl_definition*
DICTIONARY *source_ddl_dictionary*
FILTER (*expression*)
RANGE (*range_specification*)
WHERE (*where_condition*)
See "Selecting Records".

COLMAP ([USEDEFAULTS], *column_map_specification*)
COMPENSCRIBEMAPS | NOCOMPENSCRIBEMAPS
INSERTALLRECORDS
MAPID
PARTIALCOLSOK | NOPARTIALCOLSOK
SQLNAME
TARGETDEF *target_ddl_definition*
TARGETDICT *target_ddl_dictionary*
USESOURCERECLENGTH
See "Mapping Data".

KEYCOLS (*column_list*)
See "Defining Primary Key Columns".

EXITPARAM "*exitparam_string*"
See "Passing Literal Strings to User Exits".

CREATETEMPLATE *file_name*
ALTFILECHAR *num_chars*
See "Creating a Target Enscribe File".

USEALTKEY (*key_specifier*)
See "Specifying Alternate Keys".

UNMAPPEDALTFILECREATES {ALTFILEVOL | PRIMARYVOL}
See "Replicating File Create Operations for Alternate Key Files"

HANDLECOLLISIONS | NOHANDLECOLLISIONS
See "Turning Error Handling On and Off"

DETECTLOCKS
See "Locking Records".

REPERROR (*error_number*, *response*)
See "Using REPERROR".

EXCEPTIONSONLY
MAPEXCEPTION (TARGET *exception_name*, *mapping_arguments*)
See "Creating an Exceptions Statement".

GETNETWORKALTFILENAMES | IGNORENETWORKALTFILENAMES
See "Qualifying Alternate Key File Names".

PARTMAP (*source_partition_spec*, *target_partition_spec*)
Use PARTMAP to specify alternative mapping of partitions during file creation operations. For details see the PARTMAP parameter on "PARTMAP"

SHOWSYNTAX
See "Displaying a SQL Statement".

SQLEXEC (*sqlexec_clause*)
See "Performing a Query".

EVENTACTIONS (*action_options*)
See "Triggering Actions".

Selecting Records

You can select records by:

- Selecting or excluding records using FILTER.
- Selecting based on a conditional statement using WHERE.
- Selecting a subset or records using RANGE.

> **Note:**
>
> Using the RANGE option of FILE or MAP provides different capabilities than using the @RANGE function within a FILTER. And both of these are different than the RANGE option of ALTINPUT.

Selecting or Excluding Records Using FILTER

Use FILTER expressions to select or exclude data based on a numeric value. You can use a filter expression with conditional operators (such as @IF), column-conversion functions, or both. When using a FILTER expression, you can apply the filter clause to only certain record types, or specify one or more record types to omit.

If you are selecting from an Enscribe file using FILTER, you must also specify the DEF and DICTIONARY keywords.

Syntax

```
MAP source_file_name, TARGET target_file_name,
FILTER (filter_clause
[, ON INSERT | ON UPDATE| ON DELETE]
```

```
[, IGNORE INSERT | IGNORE UPDATE | IGNORE DELETE]
[, RAISEERROR error_number]
)
[, DEF source_ddl_definition]
[, DICTIONARY source_ddl_dictionary]
;
```

filter_clause
Selects records from a source `MAP` based on an expression.

`ON INSERT | ON UPDATE| ON DELETE`
Specifically limits the filter to be executed on an insert, update or delete. You can specify more than one `ON` option. For example, `ON UPDATE`, `ON DELETE` executes on updates and deletes, but not inserts.

`IGNORE INSERT | IGNORE UPDATE | IGNORE DELETE`
Ignores the specified operation. You can specify more than one `IGNORE` option.

`RAISEERROR` ***`error_number`***
Raises a user-defined error number if the filter fails. Can be used as input to the `REPERROR` parameter to invoke error handling. Make certain that the value for *`error_number`* is outside the range of error numbers that is used by the database or Oracle GoldenGate. For example: `RAISEERROR 21000`

`DEF` ***`source_ddl_definition`***
Has meaning only for Enscribe files. Use a DDL definition or record within the open dictionary. This definition describes the record that is extracted from the audit trails. You cannot specify more than one definition for any `FILE` statement.

`DICTIONARY` ***`source_ddl_dictionary`***
Points to the location of the source DDL dictionary. `DICTIONARY` establishes an Enscribe DDL dictionary to use for evaluating `WHERE`, `FILTER`, and `COLMAP` clauses. For example, `DICTIONARY $DATA5.PRODDICT` specifies a physical subvolume.
Each `DICTIONARY` entry closes any previously open dictionary, so only one dictionary is active at any given time while processing the startup parameters.

Example

The following example inserts the state `CA` if the column string matches `"CA"`.

```
MAP $PROD1.CUST.BRANCH, TARGET $DATA01.C9701.BRANCH,
DEF BRANCH-REC,
FILTER (@IF(@STREQ(STATE, "CA"), 1, 0), ON INSERT);
```

Selecting Based on a Conditional Statement

With the `WHERE` option, you can select information based on a conditional statement. If you are selecting from an Enscribe file using `WHERE`, you must also specify the `DEF` and `DICTIONARY` keywords.

Syntax

```
MAP source_file_name, TARGET target_file_name,
WHERE (where_condition)
[, DEF source_ddl_definition]
[, DICTIONARY source_ddl_dictionary]
;
```

where_condition
Selects a subset of records from a source `MAP`, based on a condition, such as `WHERE (BRANCH = "NY")`. For a list of valid operators, see Table 2-33.

`DEF` ***source_ddl_definition***
Has meaning only for Enscribe files. Use a DDL definition or record within the open dictionary. This definition describes the record that is extracted from the audit trails. You cannot specify more than one definition for any `FILE` statement.

`DICTIONARY` ***source_ddl_dictionary***
Points to the location of the source DDL dictionary. `DICTIONARY` establishes an Enscribe DDL dictionary to use for evaluating `WHERE`, `FILTER`, and `COLMAP` clauses. For example, `DICTIONARY $DATA5.PRODDICT` specifies a physical subvolume.
Each `DICTIONARY` entry closes any previously open dictionary, so only one dictionary is active at any given time while processing the startup parameters.

Table 2-33 Permissible WHERE Operators

Operator	Example
Column names	`PRODUCT_AMT`
Numeric values	`-123, 5500.123`
Literal strings enclosed in quotes	`"AUTO", "Ca"`
Column tests	`@NULL`, `@PRESENT`, `@ABSENT` (column is null, present or absent in the record). These tests are built into Oracle GoldenGate.
Comparison operators	`=, <>, >, <, >=, <=`
Conjunctive operators	`AND, OR`
Grouping parentheses	Use open and close parentheses for logical grouping of multiple elements.

Selecting a Subset of Records

Use the `RANGE` clause to select a subset of the records from Replicat's source. Unlike `WHERE`, `RANGE` does not require knowledge of the source file's structure.

Syntax

```
MAP source_file_name, TARGET target_file_name,
RANGE (x [, x, ...] of y);
```

`(x [, x, ...] of y)`
Selects a subset of records from Replicat's source, based on a condition, such as `RANGE (3 of 5)` or `RANGE (1, 3 of 5)`.

Mapping Data

Oracle GoldenGate has the following data mapping capability for the `MAP` parameters:

- Mapping columns.

- Matching source and target record lengths.

If no explicit column mapping is specified with COLMAP, Replicat determines the column map using the following rules:

- Columns with the same name are mapped to each other if the data types of the source and target are compatible.
- If the target definition has column names corresponding to special transaction values those values are mapped to the target columns.
- Column names are changed to uppercase for name comparison.
- Global rules set up with COLMATCH parameters enable different column names to be mapped by default.
- Target columns that do not correspond to any source column take default values determined by the database.

The default mapping is displayed in the report file just after the corresponding MAP entry.

Mapping Columns

Using a COLMAP clause, you can retrieve fields or columns from one record and map them to a differently structured record. This is useful, for example, when replicating data from an Enscribe file to an SQL table with similar, but not identical, fields. COLMAP selects, translates, and moves the fields you want into the new structure. When associated records are output, they are identified by the target file rather than the source to reflect the new structure of the record.

- When mapping from an Enscribe file, include either the DEF and DICTIONARY keywords, otherwise the source and target structures are assumed to be identical. DEF and DICTIONARY are required only once in the parameter file, and only when using a COLMAP clause.

  When Enscribe files without corresponding DEF parameters are encountered, source and target structures are assumed to be identical.
- When SQL tables are identified as the target, the layout of the target record after mapping is known since the SQL table structure is retrieved from the SQL catalog.

Additionally, you can match the source record length to the target record length. See "Matching Source and Target Record Lengths".

Syntax

```
MAP source_file_name, TARGET target_file_name,
COLMAP ([USEDEFAULTS], column_map_specification)
[, option];
```

column_map_specification
The column mapping expression, as in

```
(target_column = source_expression)
```

Explicitly defines a source-target column map.

target_column
The name of the target column.

source_expression
Any of the following:

- Numeric constant, such as 123
- String constant enclosed within quotes, such as "ABCD"
- The name of a source column, such as `ORD_DATE`
- An expression using an Oracle GoldenGate column-conversion function, such as `@STREXT (COL1, 1, 3)`

Example:

```
COLMAP (USEDEFAULTS,
targcol1 = srccol1,
targcol2 = srccol2,
targcol3 = srccol3)
```

TARGET *target_file_name*
Names the target file. Must be an existing Enscribe file or SQL table, and can be an active define name.

option
A DDL definition obtained by the following:

DEF *source_ddl_definition*
Has meaning only for Enscribe files. Use a DDL definition or record within the open dictionary. This definition describes the record that is extracted from the audit trails. You cannot specify more than one definition for any `MAP` statement.

NOCOLMAP
Allows the user to specify a `DEF` for filtering purposes, but prevents the columns mapping command from completing. Example:

```
MAP \PROD.$DATA06.CER1ATLF.TL*, TARGET \GGS2.$DATA10.GGSLOGS.*,
DEF TLF,
NOCOLMAP,
WHERE (TLF.HEAD.REC-TYP <> "00");
```

DICTIONARY *source_ddl_dictionary*
Points to the location of the source DDL dictionary. `DICTIONARY` establishes an Enscribe DDL dictionary to use for evaluating `WHERE`, `FILTER`, and `COLMAP` clauses. For example, `DICTIONARY $DATA5.PRODDICT` specifies a physical subvolume. Each `DICTIONARY` entry closes any previously open dictionary, so only one dictionary is active at any given time while processing the startup parameters.

TARGETDEF *target_ddl_definition*
Use `TARGETDEF` when invoking column mapping to an Enscribe file structure and the Enscribe file has not yet been specified in the parameter file, or did not have a definition associated with it.
If you assigned a definition to the target file earlier in the parameter file, you can omit `TARGETDEF`.

TARGETDICT *target_ddl_dictionary*
Points to the target DDL dictionary. Use in conjunction with `TARGETDEF` when the target definitions are in a DDL dictionary different from `DEF`. Follows the `TARGETDEF` *`targetdef`* entry, similar to:

```
MAP $vol.subvol.source, DEF sourcedef, TARGET $vol.subvol.target,
TARGETDEF targetdef, TARGETDICT $vol.subvol,
COLMAP (USEDEFAULTS
targcol1 = srccol1,
targcol2 = srccol2,
targcol3 = srccol3);
```

USEDEFAULTS
Causes Oracle GoldenGate to automatically map source and target columns that have the same name. USEDEFAULTS eliminates the need to explicitly map every source column unless the target column has a different name. This is the default unless an explicit column mapping is used.

Matching Source and Target Record Lengths

Use the USESOURCERECLENGTH option to adjust the target record length to the length of the source record. Precede the COLMAP statement with the USESOURCERECLENGTH option.

Syntax

```
USESOURCERECLENGTH
```

Example

```
USESOURCERECLENGTH
COLMAP (USEDEFAULTS,
CRD-TYP = @STRSUB (CRD-TYP, "VD", "PV"),
FIID = @STRSUB (FIID, "WA", "SFNB"),
FIID = @STRSUB (FIID, "ID", "BAID"));
```

Compressing Enscribe Records

Use COMPENSCRIBEMAPS to compress an update record after column mapping on an Enscribe target. Compressing ensures that updates to an Enscribe file after column mapping can only update the fields that were changed.

Default

```
COMPENSCRIBEMAPS
```

Syntax

```
MAP source_file_name, TARGET target_file_name
{, COMPENSCRIBEMAPS | NOCOMPENSCRIBEMAPS};
```

When Replicat is compressing an update record and the size of the column exceeds the amount of available data, a compressed fragment is produced for the available partial data. Use NOPARTIALCOLSOK to not compress an update fragment.

Default

PARTIALCOLSOK (Compress partial column values)

Syntax

```
MAP source_file_name, TARGET target_file_name,
DEF sourcedef, TARGETDEF targetdef, NOPARTIALCOLSOK;
```

Inserting All Records

Use the INSERTALLRECORDS option to apply all record types as inserts for the current map. This will create a record of all operations made to the target record instead of maintaining only the current version. This can be useful when complete transaction history is needed. See "INSERTALLRECORDS | NOINSERTALLRECORDS" for details on INSERTALLRECORDS.

Syntax

```
MAP source_file_name, TARGET target_file_name,
INSERTALLRECORDS;
```

Defining Primary Key Columns

Use the KEYCOLS option to define one or more columns of the table as a primary key for use by Oracle GoldenGate. Oracle GoldenGate uses the key to locate rows for updates and deletes and to prevent duplicate inserts.

KEYCOLS also applies to view definitions when a view is used as the file parameter. KEYCOLS has the following dependencies:

- You must use key columns when the primary key cannot be determined, or when a SYSKEY does not exist and either FORMATSQL or FORMATASCII are specified in the parameter file.
- If the same file name is specified in multiple MAP entries, only the first KEYCOLS entry takes effect.

Syntax

```
MAP source_file_name, TARGET target_file_name,
KEYCOLS (column [, column] [, ...]);
```

(*column*)
Defines a column to be used as a substitute primary key. To specify multiple columns, create a comma-delimited list as in:

```
KEYCOLS (COL1, COL2);
```

Example

Consider a relative SQL table R1 with the columns SYSKEY, CUSTOMER, BALANCE and a unique index on the CUSTOMER column. For the following transaction, you will get different results depending on whether you have specified KEYCOLS or FORMATSQL.

```
UPDATE R1 SET BALANCE = 20 WHERE CUSTOMER = "SMITH";
```

If your FILE parameter statement is simply: FILE R1; and FORMATSQL is included in the parameter file, but no KEYCOLS are specified, the output is similar to:

```
UPDATE R1 SET BALANCE = 20 WHERE SYSKEY = 1334519;
```

If instead the FILE entry is: FILE R1, KEYCOLS (CUSTOMER); the output is:

```
UPDATE R1 SET BALANCE = 20, SYSKEY = 1334519 WHERE CUSTOMER = "SMITH";
```

Passing Literal Strings to User Exits

Use EXITPARAM to pass a literal string to user exit routines whenever a record from MAP is encountered.

The string must be enclosed in double quotes and an ampersand used if it continues to additional lines. It is unlimited in size, but you must use the new function GET_EXIT_PARAM_VALUE to access values over the default of 256 bytes.

Syntax

```
MAP source_file_name, TARGET target_file_name,
EXITPARAM "exitparam_string"
```

Creating a Target Enscribe File

To deliver to an Enscribe file that does not yet exist, a file is created according to rules in a file template. Using the template file name, a file is created with the same structure, alternate keys and partitions.

The new file substitutes the target file name in the map for the template name (file name only, not subvolume), and uses the partition name, alternate key volumes and subvolumes of the template. The altkeys file names are derived from the new file name: by default, by appending 0, 1, 2 to the file name. If ALTFILECHAR is specified, a 0, 1, 2 is inserted at the indicated character.

CREATETEMPLATE is invoked upon an insert if the file does not exist. It is not invoked on file create operations, so when CREATETEMPLATE is used, file creates should not be captured. The CREATETEMPLATE option is valid for MAP statements that use wildcards or fully qualified file names. It applies only to Enscribe files.

If a file is renamed or removed after OPENTIMEOUT, the next operation on the missing file will trigger creation of a new file if the CREATETEMPLATE parameter is specified. Replicat will recognize that the create time of the target file changed and refresh the file attributes.

Syntax

```
MAP source_file_name, TARGET target_file_name,
CREATETEMPLATE file_name, ALTFILECHAR num_chars
```

Example

```
MAP $DATA2.DAT.TL*,
TARGET $DATA5.DAT.*,
CREATETEMPLATE $DATA3.GGSMASK.TLYYMMDD,
ALTFILECHAR 2;
```

Specifying Alternate Keys

Use USEALTKEY when replicating updates to Enscribe entry-sequenced or relative files, since Replicat cannot guarantee that the keys in the source and target will match. If target records can be uniquely identified using an alternate key, USEALTKEY enables Replicat to update the correct record.

Do not use this option when the unique alternate key is updated.

To ensure USEALTKEY works correctly when replicating data, turn off update compression for any entry-sequenced files that depend on this method. Turn off

compression using FUP for TMF-audited files and using the Logger configuration for non-TMF files.

Syntax

```
MAP source_file_name, TARGET target_file_name,
USEALTKEY "key_specifier"
```

"key_specifier"
The unique key identifier.

Examples

Example 1

```
USEALTKEY "TS"
```

Example 2

The following parameter file example uses an alternate key for updates to an Enscribe relative file. The first set of statements insert the record ignoring the `SYSKEY`. The second set uses the alternate key `"TM"` to locate records for updates and deletes.

```
IGNOREUPDATES
IGNOREDELETES
GETINSERTS
MAP \NY.$DATA1.PRDDAT.FILEA,
TARGET \LA.$DATA3.BKDAT.FILEA,
DEF FILEA-REC, TARGETDEF FILEA-REC,
COLMAP (USEDEFAULTS, SYSKEY = -2);
GETUPDATES
GETDELETES
IGNOREINSERTS
MAP \NY.$DATA1.PRDDAT.FILEA,
TARGET \LA.$DATA3.BKDAT.FILEA,
USEALTKEY "TM";
```

Replicating File Create Operations for Alternate Key Files

Use `UNMAPPEDALTFILECREATES` when you want to replicate file create operations for alternate key files, but have not included a `MAP` statement for the alternate key file. You can create your alternate key file one of two ways: by creating it from the alternate key file on the source system, or by creating it from the location of the primary file's volume.

> **Note:**
>
> If you have provided a `MAP` for the alternate keys, you will not require this option.

Syntax

```
MAP source_file_name, TARGET target_file_name,
UNMAPPEDALTFILECREATES {ALTFILEVOL | PRIMARYVOL}
```

ALTFILEVOL
Creates the alternate key file based on the location of the source system's alternate keys file. If this is not possible, the process abends with the -2 mapping error.

PRIMARYVOL
Attempts to create the alternate key file based on the location of the source system's alternate key file. If this is not possible, it creates an alternate key file based on the location of the primary file's volume.

Example

The following example will create an alternate key file based on the location of the file's primary volume if the source system alternate key file cannot be located.

```
MAP $DATA02.TSSOUT.ALTXX*, TARGET $DATA4.TSSIN.*, UNMAPPEDALTFILECREATES PRIMARYVOL;
```

Turning Error Handling On and Off

HANDLECOLLISIONS can be set to ignore duplicate and missing record errors for the MAP statement. NOHANDLECOLLISIONS will turn this off.

Syntax

```
MAP source_file_name, TARGET target_file_name,
HANDLECOLLISIONS;
```

Example

The following example will turn HANDLECOLLISIONS on for all of the target files included in the ORD* wildcard and off for CUSTOMERS.

```
MAP $DATA2.GGSSOU.ORD*, TARGET $DATA4.GGSTAR.*,
HANDLECOLLISIONS;
MAP $DATA2.GGSOUT.CUSTOMERS, target $DATA4.GGSIN.*,
NOHANDLECOLLISIONS;
```

Locking Records

For Enscribe files, DETECTLOCKS causes Replicat to issue a SETMODE 4 to reject a lock request with an error 73. For SQL tables, it causes Replicat to issue the "CONTROL TABLE RETURN IF LOCKED" statement.

Setting a REPERROR clause allows Replicat to retry a locked record a specified number of times.

> **Note:**
>
> Use of DETECTLOCKS could cause an increase in the number of error 73s reported. There can be a lag between when TM/MP tells the application that the transaction has been committed and when DP2 actually releases the locks. Without the DETECTLOCKS parameter set, Oracle GoldenGate waits for DP2 to release its locks before continuing. With the DETECTLOCKS parameter active, Oracle GoldenGate returns an error 73 when it is first encountered. You can use REPERROR 73 to address these errors when they occur. Before implementing DETECTLOCKS review your processing needs to determine the best solution.

Syntax

```
MAP source_file_name, TARGET target_file_name, DETECTLOCKS
```

Using REPERROR

Use REPERROR to specify an error and a response that together control how Replicat responds to the error when executing the MAP statement. You can use REPERROR at the MAP level to override and supplement global error handling rules set with the REPERROR parameter (see "REPERROR"). Multiple REPERROR statements can be applied to the same MAP statement to enable automatic, comprehensive management of errors and interruption-free replication processing.

Syntax

```
MAP source_file_name, TARGET target_file_name,
REPERROR (error_number, response) [, REPERROR (error_number, response)] [, ...];
```

Refer to the REPERROR parameter on "REPERROR" for details on the *error_number* and *response* specifications. Note that RESET is not a valid option for REPERROR used under MAP.

Examples

The following examples show different ways that REPERROR can be used in a MAP statement in conjunction with a global REPERROR statement.

Example 1
In the following example, when *error_1* occurs for the first MAP statement, the action is *response_2*, not *response_1*, because an override was specified. However, if an *error_1* occurs for the second MAP statement, the response is *response_1*, the global response. The response for *error_2* is *response_3*, which is MAP-specific.

```
REPLICAT group_name
REPERROR (error_1, response_1)
MAP source_1, TARGET target_1, REPERROR (error_1, response_2);
MAP source_2, TARGET target_2, REPERROR (error_2, response_3);
```

Example 2
In the following example, when replicating from src1 to src2, all errors and actions (1-3) apply, because all REPERROR statements address different errors (there are no MAP-specific overrides).

```
REPLICAT group_name
REPERROR (error_1, response_1)
MAP source_1, TARGET target_1, REPERROR (error_2, response_2),
REPERROR (error_3, response_3);
```

Example 3
In the following example, if error1 occurs for the first MAP statement, the action is response2. For the second one it is response1 (the global response), and for the third one it is response4 (because of the second REPERROR statement). A global REPERROR statement applies to all MAP statements that follow it in the parameter file until another REPERROR statement starts new rules.

```
REPLICAT group_name
REPERROR (error_1, response_1)
```

```
MAP source_1, TARGET target_1, REPERROR (error_1, response_2);
MAP source_2, TARGET target_2, REPERROR (error_2, response_3);
REPERROR (error_1, response_4)
MAP source_2, TARGET target_2, REPERROR (error_3, response_3);
```

Creating an Exceptions Statement

Errors that have REPERROR set to EXCEPTION can be captured to an exception file using either EXCEPTIONSONLY or MAPEXCEPTION.

Using EXCEPTIONSONLY

EXCEPTIONSONLY specifies that the current map is executed only when an error occurred for the last record processed in the preceding map. You must set REPERROR to EXCEPTION for the error that occurred, and the exception map must specify the same source table as the map in error. The exception map must follow the map in error in the parameter file.

Syntax

```
MAP source_file_name, TARGET target_file_name,
EXCEPTIONSONLY;
```

> **Note:**
>
> The source and target file names in the preceding MAP statement (the one the EXCEPTIONSONLY applies to) cannot include wildcards.

Using MAPEXCEPTION

MAPEXCEPTION specifies a target file for exceptions in processing the source and target files of the MAP. The source and target file names can include wildcards and all exceptions that apply to the file set are written to the *exception_file_name*. Any valid mapping arguments, such as COLMAP or KEYCOL, can be included.

Syntax

```
MAP source_file_name, TARGET target_file_name,
MAPEXCEPTION (TARGET exception_file_name,
mapping_arguments);
```

exception_file_name
Identifies the target file for errors that have been identified as EXCEPTION with REPERROR.

mapping_arguments
Any valid mapping argument that can be used with the MAP statement.

Example

This example uses wildcarded source and target file names. Exceptions that occur when processing any file on $DATA02.ACCT with a name beginning with TRX will be captured to $DATA08.ACCT.OLDTRX using the designated mapping.

```
MAP $DATA04.ACCT.TRX*, TARGET $DATA05.ACCT.*,
MAPEXCEPTION (TARGET $DATA08.ACCT.OLDTRX,
   COLMAP (USEDEFAULTS,
```

```
   ACCT-NO = ACCT-NO,
   OPTYPE = @GETENV ("LASTERR", "OPTYPE"),
   DBERR = @GETENV ("LASTERR", "DBERRNUM"),
   DBERRMSG = @GETENV ("LASTERR", "DBERRMSG")
   )
);
```

Qualifying Alternate Key File Names

`GETNETWORKALTFILENAMES` lets the file system qualify the alternate key file names with the local node name. Use `IGNORENETWORKALTFILENAMES` to tell the file system not to qualify the alternate key file names with the local node name. This parameter can also be toggled around `MAP` statements.

Default

```
GETNETWORKALTFILENAMES
```

Syntax

```
MAP source_file_name, TARGET target_file_name,
[GETNETWORKFILENAMES | IGNORENETWORKALTFILENAMES];
```

Displaying a SQL Statement

Use `SHOWSYNTAX` to display a SQL statement before it is executed. The default is to display SQL statement text in the report file.

Syntax

```
MAP source_file_name, TARGET target_file_name, SHOWSYNTAX;
```

Example

```
REPSQLLOG $DATA.SQLLOG.JSR01
MAP $DATA.SOURCE.ACCOUNT, TARGET $DATA.TARGET.ACCOUNT,
SHOWSYNTAX,
MAPID "My Mapid",
COLMAP (USEDEFAULTS);
```

The `MAPID` is displayed along with the SQL statement. Use the `MAPID` option to help in complicated mapping situations that require conditional mapping using a `WHERE` clause. If `MAPID` is not specified, a default ID of the form `"MAP"` is built with a sequential number indicating the `COLMAP` for the file. The `MAPID` text can be up to 32 bytes long. If it contains spaces, it must be enclosed in either single or double quotes.

An optional Replicat parameter `REPSQLLOG` redirects the output to a separate log file and keeps it out of the Replicat report file. See "REPSQLLOG".

Performing a Query

Use `SQLEXEC` to perform a SQL Query when processing a record for a SQL/MP table. `SQLEXEC` enables Oracle GoldenGate to communicate directly with the database to perform any query that SQL supports. The database function can be part of the synchronization process, such as retrieving values for column conversion, or it can be independent of extracting or replicating data.

> **Note:**
>
> This feature is not available for Enscribe files. SQLEXEC queries should not be used to change a value in the primary key column. The primary key value is passed from Extract to Replicat so Replicat can perform further update/delete operations. If Replicat does not know the primary key value, these operations cannot be completed.

By using SQLEXEC within multiple FILE or MAP statements, you can create different rules for different tables; these rules can be as simple or as complex as needed. A query that is executed can accept input parameters from source or target rows and pass output parameters.

In the following example, SQLEXEC runs a select statement, extracting the timestamp from the target table, then filters out records as needed.

```
MAP $DATA1.SQLDAT.ORDERS, TARGET $DATA1.MASTER.ORDERS,
SQLEXEC (ID check,
QUERY " SELECT TIMESTAMP FROM $DATA1.SQLDAT.ORDERS "
" WHERE PKCOL = ?P1 ", PARAMS (P1 = PKCOL), ERROR REPORT);
```

A SQLEXEC statement expects legal SQL syntax for the database being affected. Refer to the SQL for NonStop reference guide for permissible SQL syntax.

> **Note:**
>
> If a SQLEXEC query fails, the Extract or Replicat process will exit. As such, you must structure your query correctly.

Syntax

```
MAP source_file_name, TARGET target_file_name,
SQLEXEC (
ID logical_name,
QUERY "sql_query",
{PARAMS param_spec | NOPARAMS}
[, AFTERFILTER | BEFOREFILTER]
[, DBOP]
[, EXEC frequency]
[, MAXVARCHARLEN bytes]
[, PARAMBUFSIZE num_bytes]
[, TRACE option]
[, ALLPARAMS option]
[, ERROR action]
[, option] [, ...]
)
```

ID *logical_name*
Defines a logical name for the query. A logical name is required in order to extract values from the query results. ID *logical_name* references the column values returned by the query.

QUERY "*sql_query*"
Specifies the SQL query syntax to execute against the database. The query must be valid, standard query statement for the database against which it is being executed. It can either return results with a `SELECT` statement or update the database with an `INSERT`, `UPDATE`, or `DELETE` statement.
For any query that produces output with a `SELECT` statement, only the first row returned by the `SELECT` is processed. Do not specify an `"INTO..."` clause for any `SELECT` statements.
Enclose the query within quotes. For a multi-line query, use quotes on each line. To ensure success, place a space after each begin quote and each end quote, or at least before the end quote. For example, in the following, there are spaces before the words `select` and `where` and after the words `ggs_notify` and `?p1`."

```
SQLEXEC (
ID ggs, ON UPDATES, ON INSERTS,
QUERY " select notified from $DATA1.SQLDAT.NOTIFY "
" where account_no = ?p1 ",
PARAMS (p1 = account_no)
)
```

PARAMS *param_spec* | NOPARAMS
Defines whether the query accepts parameters. One of these options must be used.

AFTERFILTER | BEFOREFILTER
Specifies when to execute the query in relation to a `FILTER` clause. `AFTERFILTER` executes after the filter and enables you to skip the overhead of executing the SQL unless the filter is successful. This is the default. `BEFOREFILTER` executes before the filter and enables you to use the results of the procedure or query in the filter.

DBOP
Commits `INSERT`, `UPDATE`, `DELETE`, and `SELECT` statements executed within the query. Otherwise, they could potentially be rolled back. Oracle GoldenGate issues the commit within the same transaction boundaries as the source transaction. Use caution: any changes that are committed by the procedure can result in overwriting existing data.

EXEC *frequency*
Controls the frequency with which a query executes and how long the results are considered valid, if extracting output parameters. Takes one of the following arguments:

MAP
Executes the query once for each source-target table map for which it is specified. `MAP` renders the results invalid for any subsequent maps that have the same source table. For example, if a source table is being synchronized with more than one target table, the results would only be valid for the first source-target map. `MAP` is the default.

ONCE
Executes the query once during the course of an Oracle GoldenGate run, upon the first invocation of the associated `FILE` or `MAP` statement. The results remain valid for as long as the process remains running.

TRANSACTION
Executes the query once per source transaction. The results remain valid for all operations of the transaction.

SOURCEROW
Executes the query once per source row operation. Use this option when you are synchronizing a source table with more than one target table, so that the results of the procedure or query are invoked for each source-target mapping.

MAXVARCHARLEN ***bytes***
Specifies the maximum length allocated for any output parameter in a query. Beyond this maximum, output values are truncated. The default is 255 bytes without an explicit `MAXVARCHARLEN` clause.

PARAMBUFSIZE ***num_bytes***
Specifies the maximum size of the memory buffer that stores parameter information, including both input and output parameters. Oracle GoldenGate issues a warning whenever the memory allocated for parameters is within 500 bytes of the maximum. The default is 10,000 bytes without an explicit `PARAMBUFSIZE` clause.

TRACE ***option***
Takes one of the following arguments:

TRACE ALL
Writes the input and output parameters of each invocation of a query to the report file.

TRACE ERROR
Writes parameters to the report file only if an error occurs.

ALLPARAMS ***option***
Takes one of the following arguments:

ALLPARAMS REQUIRED
Indicates that all parameters must be present for the queries to execute.

ALLPARAMS OPTIONAL
Allows the query to execute without all parameters being present.

ERROR ***action***
Requires one of the following arguments:

ERROR IGNORE
Database error is ignored and processing continues.

ERROR REPORT
Database error is written to a report.

ERROR RAISE
Database error is handled just as a table replication error.

ERROR FINAL
Database error is handled as a table replication error, but does not process any additional queries.

ERROR FATAL
Database processing abends.

Triggering Actions

EVENTACTIONS can be used to trigger an event based on a file or table receiving a DML record. The event actions will take place after mapping and either before or after the record is processed depending on the action.

Syntax

```
MAP source_file, TARGET target_file
EVENTACTIONS ([VERBOSE]
  [, IGNORE | DISCARD]
  [, TACLCMD ("CMD_file_details") | TACLCMD (CMD $1, VAR $1 = value)]
  [, EMS WARN | INFO]
  [, CHECKPOINT {BEFORE | AFTER | BOTH}]
  [, REPORT]
  [, STOP]
  [, SUSPEND])
```

VERBOSE
Writes details to the report for each event as it is processed

IGNORE
Skips the record. IGNORE and DISCARD are incompatible, so only one of these options can be used.

DISCARD
Discards the record. IGNORE and DISCARD are incompatible, so only one of these options can be used.

TACLCMD
Executes a user-defined system command. Valid TACLCMD file commands are PURGE, PURGEDATA, RUN, RENAME, OBEY, and FUP. Non-file values can be obtained by SQLEXEC, @GETENV(), or Column Data.

EMS
Writes either INFO or WARN messages to EMS.

CHECKPOINT
Checkpoints its position BEFORE the record is processed, AFTER the record is processed or BOTH.

REPORT
Writes the current statistics to the report file.

STOP
Stops the process.

SUSPEND
Suspends the process until it is resumed by a command from GGSCI.

```
MAP $DATA.SLS.PLR, TARGET $DATA3.SLS.PLA
  EVENTACTIONS (VERBOSE, TACLCMD "RENAME $DATA.SLS.PLR, $DATA.SLS.T4M")
```

The following example sets VAR options using parameters from SQLEXEC:

```
EXTRACT EXMPAB
SETENV FETCHTBL=$DATA02.TOJ06n.ENVENTAB
```

```
MAP$data02.TOJOUT.EMARKER, TARGET $data03.TOJOUT.EMARAKER,
  COLMAP (COL1 = COL1, SYSKEY=SYSKEY);SQLEXEC (ID getprog,
  QUERY "SELECT * FROM ?FETCHTBL"
  WHERE "table_name =?T1",
PARAMS(T1=@GETEMV("GGHEADER","TABLENAME")), ERROR REORT),
EVENTACTIONS (VERBOSE,EMS WARN,REPORT,CHECKPOINT BEFORE,
  TACLCMD("RUN $1/CPU $3,PRI $4, NAME $5/-f$2",
    var $1=GETPROG.program_name,
    var $2=GETPROG.file_name,
    var $3=GETPROG.cpu,
    var $4=GETPROG.pri,
    var $5=GETPROG.process),
  DISCARD);
```

Examples

The following example writes to the report and executes a `TACLCMD` to rename the file:

MAPEXCLUDE

Valid for

Replicat

Description

Use `MAPEXCLUDE` to exclude files or file sets from a wildcard map list. `MAPEXCLUDE` operates globally within the Replicat parameter file, therefore a file is excluded from mapping if it is specified in any `MAPEXCLUDE` entry.

Syntax

```
MAPEXCLUDE file_name
```

file_name
The file to exclude from mapping. You can use wildcards.

Example

The following example maps file names that begin with `A`, except for files in subvolume `BADSUB`, files that begin with `AB` and the file `$D1.BAD.A123`.

```
MAP $D1.*.A*, TARGET $D2.*.*;
MAPEXCLUDE $D1.BADSUB.*
MAPEXCLUDE $D1.*.AB*
MAPEXCLUDE $D1.BAD.A123
```

MAXABENDRESTARTS

Valid for

`GLOBALS`, Manager

Description

Use `MAXABENDRESTARTS` to specify the number of times Manager tries to restart a process that stopped due to either of the following conditions:

- The process ended abnormally.
- The process fails because it cannot write to disk.

Restart counts are initialized every `RESTARTINTERVAL` minutes.

By default, restarts are attempted each minute. If they need to be attempted at a different interval, use `AUTORESTART` with the `WAITMINUTES` option instead.

If `AUTORESTART` with `RETRIES` is used, `MAXABENDRESTARTS` is ignored.

If neither `MAXABENDRESTARTS` nor `AUTORESTART RETRIES` are specified and the process has been restarted and failed two times, Manager will stop trying to restart the process.

Default

2

Syntax

```
MAXABENDRESTARTS count
```

count
The number of times Manager tries to restart a process. If you do not want Manager to restart an operation, set *count* to zero.

MAXDISCARDRECS

Valid for

Replicat

Description

Use `MAXDISCARDRECS` to limit the number of errors reported to the discard file per map. Limiting the number of errors reported can reveal the types of errors being experienced without causing Replicat to terminate due to full discard files.

Syntax

```
MAXDISCARDRECS num_recs
```

num_recs
The number of records to discard.

MAXETCHECKPOINTSECS

Valid for

Replicat

Description

Use `MAXETCHECKPOINTSECS` to specify the maximum amount of time that Replicat waits before writing a checkpoint. A checkpoint saves the state of processing and is used for recovery in the event of an application failure. The length of time between checkpoints defines the amount of data that may need to be reprocessed from the data source in the event of an application failure.

Reducing MAXETCHECKPOINTSECS prevents Replicat from having to read too far back into the data source if a failure occurs. However, since checkpoints commit data to disk, it could cause performance degradation because data is written to disk more frequently.

Increasing MAXETCHECKPOINTSECS reduces the number of checkpoints, which might improve performance, but it requires Replicat to read further back into the data source in the event of a failure. To make that data available, prevent the logs containing it from being removed or overwritten.

Avoid using MAXETCHECKPOINTSECS unless directed to do so by Oracle GoldenGate Technical Support.

Default

10

Syntax

```
MAXETCHECKPOINTSECS seconds
```

seconds
The number of seconds.

MAXTRANSMEM

Valid for

Extract

Description

Use MAXTRANSMEM to control the maximum amount of memory allocated for a transaction. If a transaction exceeds the maximum memory value, Extract only tracks the checkpoint of the begin transaction until the commit transaction is found. Once a commit is found, Extract returns and extracts data starting from the begin checkpoint. Do not set MAXTRANSMEM to an arbitrarily high number, since doing so leads to excessive memory consumption.

Default

5000000

Syntax

```
MAXTRANSMEM megabytes
```

megabytes
The number of megabytes. The minimum value is 2000000; the maximum value is 100000000.

Example

```
MAXTRANSMEM 5000000
```

MAXTRANSOPS

Valid for

Replicat

Description

Use `MAXTRANSOPS` to limit the number of I/Os in a transaction. Use this parameter to reduce the number of record locks required on the target database for large transactions. `MAXTRANSOPS` splits large transactions into smaller portions and can reduce transaction log requirements. Use `MAXTRANSOPS` to alleviate Guardian error 35 caused by exceeding the lock limit when performing Enscribe replication.

Syntax

```
MAXTRANSOPS op_count
```

op_count
The maximum number of operations for a single transaction.

MAXWILDCARDENTRIES

Valid for

Extract, Replicat

Description

Use `MAXWILDCARDENTRIES` to set the initial allocation for the number of wildcard entries. The default is 100 if nothing is set. Once this initial `MAXWILDCARDENTRIES` allocation is exhausted, the program will allocate an additional 100 entries each time it needs more.

You can override the default with the `MAXWILDCARDENTRIES` parameter in the Extract or Replicat parameter files.

Default

100

Syntax

```
MAXWILDCARDENTRIES number
```

number
The number of entries to be allocated initially for wildcards. It can be any number between 100 and 32,780.

NETWORKCHECKPOINTS

Valid for

Replicat

Description

NETWORKCHECKPOINTS specifies local checkpoint updates for file partitions residing on remote nodes. These updates allow the identification of replicated data in a bi-directional environment with data partitioned across nodes. The following processing occurs:

- As Replicat resolves MAP entries, it checks the partition list for partitions residing on remote nodes.
- When one is found, Replicat checks the GLOBALS parameter file for the location of the Oracle GoldenGate environment for that node.
- Replicat then uses this node, volume and subvolume to identify the REPCTXT to add to a list of checkpoint files that will be updated.

By default Replicat will set NETWORKCHECKPOINTS whenever it finds that files or partitions are on remote nodes.

Default

Checkpoint files that are found on the partition's remote node are updated.

Syntax

```
NETWORKCHECKPOINTS
[, MAXRETRIES number]
[, DELAYSECS number | DELAYCSECS number]
```

MAXRETRIES *number*
Sets the maximum number of times Replicat will retry EXPAND related errors 60, 248, 249, and 250. The default is 3 and the maximum is 100 retries.

DELAYSECS *number*
Sets the number of seconds that Replicat will wait between tries. The default is 5 seconds and the maximum is 60 seconds.

DELAYCSECS *number*
Sets the number of centiseconds that Replicat will wait between tries.

NOTSTOPPABLE

Valid for

Logger

Description

NOTSTOPPABLE enables Logger to be stopped only with the GGSCI STOP LOGGER command. To use this parameter, Logger must be run under SUPER group authority. This parameter applies only to the current log.

Syntax

```
NOTSTOPPABLE
```

NUMEXTRACTS

Valid for

Extract

Description

Use `NUMEXTRACTS` to override the default number of Oracle GoldenGate trails or files in the trail, that can be specified in the parameter file. When exceeding the default, a small amount of additional memory is consumed by Extract.

Default

50

Syntax

`NUMEXTRACTS` *number*

number
The number of Oracle GoldenGate trails.

NUMFILES

Valid for

Extract, Replicat

Description

Use `NUMFILES` to control the initial number of memory structures allocated to contain information about source and target tables specified in `TABLE` or `MAP` statements. `NUMFILES` must occur before any `TABLE` or `MAP` entries to have any effect.

To control the number of additional memory structures that are allocated dynamically once the `NUMFILES` value is reached, use the `ALLOCFILES` parameter (see "ALLOCFILES").

The default values should be sufficient for both `NUMFILES` and `ALLOCFILES`, because memory is allocated by the process as needed, system resources permitting.

Default

1000

Syntax

`NUMFILES` *num_structures*

num_structures
The number of memory structures to be allocated. Do not set `NUMFILES` to an arbitrarily high number, or memory will be consumed unnecessarily. Oracle GoldenGate memory supports up to two million tables.

Example

```
NUMFILES 4000
```

OBEY

Valid for

Extract, Replicat

Description

Use `OBEY` to retrieve parameters from a file other than the current parameter file, then return processing to current parameter file.

Syntax

```
OBEY file_name
```

file_name
The name of the parameter file to obey.

> **Note:**
>
> The file to be obeyed (*file_name*) cannot contain any `OBEY` parameters. A file that is used as an obey file cannot call another obey file.

OLDGROUPNAMING

Valid for

`GLOBALS`

Description

Use `OLDGROUPNAMING` to specify whether new group naming rules are enforced. New group naming limits Extract and Replicat group names to seven characters. Old group naming allows group names of up to ten characters.

Default

`NO`

Syntax

```
OLDGROUPNAMING {YES | NO}
```

OMITAUDITGAPCHECK

Valid for

Extract

Description

Use `OMITAUDITGAPCHECK` to enable Extract to continue processing even when a gap is detected between the oldest required audit trail and the current trail. By default, Extract checks for the presence of required TMF audit trails at the beginning of processing. When the audit trials are not available at the beginning of processing, but will be available later, `OMITAUDITGAPCHECK` can be specified to avoid terminating abnormally at startup.

Syntax

```
OMITAUDITGAPCHECK
```

OPENTIMEOUT | OPENTIMEOUTMINUTES

Valid for

Extract, Replicat

Description

Use `OPENTIMEOUT` or `OPENTIMEOUTMINUTES` to control when individual Enscribe files are closed after a period of inactivity.

Replicat opens target Enscribe files when the first replicated database change is received and the files remain open to receive additional records.

Extract opens source Enscribe files for `FETCHCOMPS` and `FETCHLASTIMAGE` and the files remain open.

If there is no activity for these opened files, the file closes when the `OPENTIMEOUT` value is reached. Low values can have a negative effect on performance

Default

60 minutes

Syntax

```
OPENTIMEOUTMINUTES minutes
```

minutes
The inactivity threshold in minutes.

OPENWARNINGS

Valid for

Replicat

Description

Use `OPENWARNINGS` to control how Replicat responds when Guardian issues a warning during a target Enscribe file open. This can happen, for example, when an alternate key file for the target file is unavailable (Guardian error 4) or a secondary partition is unavailable (error 3).

Default

WARN

Syntax

```
OPENWARNINGS {WARN | IGNORE | ABEND}
```

WARN
Issues a warning but continues.

IGNORE
Continues without warning.

ABEND
Quits processing with an irrecoverable error message.

OVERRIDEDUPS | NOOVERRIDEDUPS

Valid for

Replicat

Description

Use OVERRIDEDUPS to direct Replicat to always apply the record it is currently processing. When NOOVERRIDEDUPS is in effect, a duplicate error is returned and the transaction may abort.

Duplicate errors can be caused when multiple sources insert the same record into a single target. If both of these operations are replicated to the same target, one will fail because each has the same primary key.

NOOVERRIDEDUPS remains in effect until OVERRIDEDUPS is specified. OVERRIDEDUPS remains in effect for all maps specified below it until NOOVERRIDEDUPS is entered.

Records may not be processed in chronological order across Replicat processes. To bypass duplicate records without causing Replicat to terminate abnormally, specify either of the following REPERROR parameters. Even so, when duplicate errors occur, records are output to the discard file.

```
REPERROR (10, IGNORE)
REPERROR (-8227, IGNORE)
```

OVERRIDEDUPS is automatically in effect when HANDLECOLLISIONS is specified.

Default

NOOVERRIDEDUPS

Syntax

```
OVERRIDEDUPS | NOOVERRIDEDUPS
```

PARTMAP

Valid for

Replicat

Description

Use PARTMAP to specify alternative mapping of partitions during file creation operations. PARTMAP affects only the creation of files having secondary partitions. You can specify wildcards to map sets of disk drives on the source system to similar but not equal sets on the backup system. Specific entries (those without wildcards) take precedence when applicable over wildcard entries.

When PARTMAP is added as an independent parameter it turns on that alternative mapping only for MAP statements that follow it.

PARTMAP can be also be used as an option to a MAP statement. See "MAP".

Syntax

```
PARTMAP source_partition_spec, target_partition_spec
```

source_partition_spec
The source partition specification of a volume name or a partial volume name followed by an asterisk.

target_partition_spec
The target partition specification of a volume name or a partial volume name followed by an asterisk.

Example

To map secondary partitions created on $DATA1 to $D15:

```
PARTMAP $DATA1, $D15
```

PARAMCHECK

Valid for

GLOBALS

Description

Use PARAMCHECK to specify whether Extract and Replicat check to ensure that the correct parameter file and process name have been specified at startup.

Syntax

```
PARAMCHECK {YES | NO}
```

PASSTHRU | NOPASSTHRU

Valid for

Extract

Description

Use PASSTHRU and NOPASSTHRU to control whether a data-pump Extract processes files in pass-through mode or normal mode. In pass-through mode, the Extract process does not look up file and table definitions, either from the database or from a data-definitions file. In normal mode the definitions are used to perform mapping and conversion functions.

Using pass-through mode, you can cascade the captured data to a data pump on an intermediary system that has no database installed on it. Source and target file names and structures must be identical; no filtering, column mapping, SQLEXEC functions, transformation, or other functions requiring data manipulation or translation can be used.

You can use PASSTHRU to allow Extract to move data after a file is purged or renamed. By default, Extract does not output data for purged or renamed source files, because no definition is available for the file. However, there are instances when this is appropriate, as when Extract distributes data created by Logger to other systems or acts as a data pump.

To ensure your trails process correctly, you must set the PASSTHRU parameter before you set your EXTTRAIL or RMTTRAIL parameters. Later in the parameter file, you can apply NOPASSTHRU then set up your where clauses or filters on files that require them. Use PASSTHRU only when Extract is distributing data already in a Oracle GoldenGate trail (LOGTRAILSOURCE/EXTTRAILSOURCE). Do not use PASSTHRU in combination with CUSEREXIT or COBOLUSEREXIT.

Enter PASSTHRU above any FILE entries that may fit this description.

Default

NOPASSTHRU

Syntax

```
PASSTHRU | NOPASSTHRU
```

Example

The following example passes through all data from $DATA.DAT even if the source file no longer exists. Only account codes less than 100 are distributed from the ACCOUNT file.

```
EXTTRAIL \XYZ.$DATA3.GGSDAT.ET
PASSTHRU
FILE $DATA1.DAT.*;
NOPASSTHRU
FILE $DATA2.DAT2.ACCOUNT, DEF ACCOUNT-REC,
    WHERE (ACCOUNT-CODE < 100);
```

PORT

Valid for

Manager

Description

For remote processes to request dynamic services, such as creating Collector processes, Manager must listen on a designated TCP/IP port. This port is defined in the parameter files of Extract groups. To specify a dynamic port, enter the `RMTHOST` parameter with its `MGRPORT` option in the Extract parameter file.

The default TCP/IP process for Manager is `$ZTC0`. To change the default process, before you start or restart the Manager process:

- Set the `TCPIPPROCESSNAME` parameter in the Manager parameter file. See "TCPIPPROCESSNAME" for more information.
- Set the `DEFINE =TCPIP^PROCESS^NAME` to the process you want. For example:

  ```
  ADD DEFINE =TCPIP^PROCESS^NAME, FILE $ZTC3
  ```

Default

None (Do not listen on a port if none is specified)

Syntax

```
PORT port_number
```

port_number
The port number.

POSITIONFIRSTRECORD

Valid for

Extract

Description

Use `POSITIONFIRSTRECORD` for direct read file extraction only. `POSITIONFIRSTRECORD` positions Extract at the beginning of the input file to process all records again.

Use this parameter only when you are trickling data throughout the day to the target system with `RMTBATCH` and the records are variable length (for example, the ACI BASE24 Super files).

If you specify `SYSKEYCONVERT`, Extract turns on `POSITIONFIRSTRECORD`.

Syntax

```
POSITIONFIRSTRECORD
```

PRIORITY

Valid for

Logger

Description

`PRIORITY` indicates the NonStop operating system priority for the current log process. Run Logger at a significantly high priority to ensure that it can keep up with the application. For example, `PRIORITY 190` sets priority at 190.

Default

Same priority as the Manager process (default of 190)

Syntax

`PRIORITY` *`priority`*

`priority`
The priority level.

PURGEDATAALTFILES | NOPURGEDATAALTFILE

Valid for

Replicat

Description

Use `PURGEDATAALTFILES` to purge data on the alternate key files when an Enscribe `PURGEDATA` is received for the primary file. To not purge the data, specify `NOPURGEDATAALTFILES`.

Default

`PURGEDATAALTFILES`

Syntax

```
PURGEDATAALTFILES | NOPURGEDATAALTFILES
```

PURGEOLDEXTRACTS for Extract and Replicat

Valid for

Extract and Replicat

Description

Use `PURGEOLDEXTRACTS` in an Extract or Replicat parameter file to delete old trail files whenever Oracle GoldenGate starts processing from a new one. Using `PURGEOLDEXTRACTS` conserves disk space by preventing the accumulation of trail files.

Purging by Extract is appropriate if the process is a data pump. After the data is sent to the target system, the files can be purged. Otherwise, purging would ordinarily be done by Replicat.

Do not use `PURGEOLDEXTRACTS` in an Extract or Replicat parameter file if more than one Extract or Replicat is reading the same set of trail files. If multiple processes are reading the same files, one process could purge a file before another is finished with it. In this case use `PURGEOLDEXTRACTS` in the Manager. As a general rule, letting the Manager handle `PURGEOLDEXTRACTS` is preferred for all Oracle GoldenGate configurations because it provides centralized management of your trail files.

Default

Purge the trail file when moving to the next file in the sequence.

Syntax

```
PURGEOLDEXTRACTS
```

PURGEOLDEXTRACTS for Manager

Valid for

Manager

Description

Use `PURGEOLDEXTRACTS` in a Manager parameter file to purge trail files when Oracle GoldenGate has finished processing them. By using `PURGEOLDEXTRACTS` as a Manager parameter, you can use options that are not available with the Extract or Replicat version. Allowing Manager to control the purging helps to ensure that no file is purged until all groups are finished with it.

To purge trail files, you can use the following rules:

- Purge if all processes are finished with a file as indicated by checkpoints. Use the `USECHECKPOINTS` option. (This is the default.)
- `MINKEEP` rules set the time or number of files to keep. Specify only one of the following three parameters.
  - Keep files or rows for at least a specified number of hours or days. Use the `MINKEEPHOURS` and `MINKEEPDAYS` options.
  - Keep at least *n* files including the active file. Use the `MINKEEPFILES` option.

The Manager process determines which files to purge based on Extract and Replicat processes configured on the local system. If at least one process reads a trail file, Manager applies the specified rules; otherwise, the rules do not take effect.

Refer to the information on managing trails in the Recommendations for Managing Trail Purgesfor recommendations on using the rules.

Default

None (do not purge)

Syntax

```
PURGEOLDEXTRACTS trail_name
[, USECHECKPOINTS | NOUSECHECKPOINTS]
[, min_rule]
```

trail_name
The trail to purge. Use the fully qualified name.

USECHECKPOINTS
Purges after all Extract or Replicat processes are finished with the file as indicated by checkpoints.

NOUSECHECKPOINTS
Allows purging without considering checkpoints, based on keeping a minimum of one file (if no `MINKEEP` rule is used) or the number of files specified with a `MINKEEP` rule.

min_rule
Use only one of the following to set rules for the minimum amount of time to keep data.

MINKEEPHOURS ***num***
Keeps an unmodified file for at least the specified number of hours.

MINKEEPDAYS ***num***
Keeps an unmodified file for at least the specified number of days.

MINKEEPFILES ***num***
Keeps at least *num* unmodified files, including the file being considered for purging.

Examples

Example 1
The following example purges all files in the `\NY.$DATA3.GGSDAT` subvolume after checkpoints indicate that they no longer are needed, but it keeps the files at least 3 days.

```
PURGEOLDEXTRACTS \NY.$DATA3.GGSDAT.*, USECHECKPOINTS, MINKEEPDAYS 3
```

Example 2
Trail files `AA000000`, `AA000001`, and `AA000002` exist. Replicat has been down for four hours and has not completed processing. The Manager parameters include:

```
PURGEOLDEXTRACTS /ggs/dirdat/AA*, NOUSECHECKPOINTS, MINKEEPHOURS 2
```

Result: All trail files will be purged because the minimums have been met.

PURGEOLDTASKS

Valid for

Manager

Description

Use PURGEOLDTASKS to purge Extract and Replicat tasks after a specific amount of time or when they have stopped gracefully.

You can indicate when to delete a task according to the following rules:

- The task was last started a specific number of days or hours ago. If the task never was started, then its creation time is used as the basis for applying the rules.
- The task stopped gracefully or never was started. This rule takes precedence over the time the task was last started. Use this rule to prevent abnormally terminated tasks from being purged.

Default

None

Syntax

```
PURGEOLDTASKS process_type group_name [, purge_option]
```

process_type
Valid values:

- EXTRACT
- REPLICAT
- ER (for both processes)

group_name
The group name or a wildcard to specify multiple groups.

purge_option
Purges if the task has not been updated for a specific number of hours or days.
Valid values:

- AFTER *number* DAYS
- AFTER *number* HOURS

USESTOPSTATUS
Purges if the task was stopped gracefully or never was started.

Example

The following example deletes Extract tasks updated at least three days ago, and it deletes the INITREP Replicat task if it stopped gracefully and was updated at least two hours ago.

```
PURGEOLDTASKS EXTRACT *, AFTER 3 DAYS
PURGEOLDTASKS REP INITREP, AFTER 2 HOURS, USESTOPSTATUS
```

PURGERESTORE | NOPURGERESTORE

Valid for

GLOBALS

Description

Use PURGERESTORE to tell Extract to purge audit trail files that are restored from tape after processing them. NOPURGERESTORE leaves restored audit files on disk.

Default

PURGERESTORE

Syntax

```
PURGERESTORE | NOPURGERESTORE
```

READER

Valid for

Coordinator

Description

Use READER to trigger the creation of a Reader process to monitor a local trail and communicate with the system Coordinator. The HOST parameter from GLOBALS file will be used to determine the location of the object to be used. Typically this will be the installation location for Oracle GoldenGate, but if it varies the PROGRAM option can be used to point to another object.

Syntax

```
READER EXTTRAIL trail_name
[, PROCESS process_name],
[, CPU primary_cpu]
[, BACKUPCPU cpu]
[, PRI priority]
[, PROGRAM program_name]
```

EXTTRAIL *trail_name*
The fully qualified identifier of the trail that will be monitored by the Reader. This is a required entry.

PROCESS *process_name*
The process name of the associated Reader.
Oracle GoldenGate recommends you allow the process name to be generated by the system. However, if you must specify an alternative process, PROCESS *process_name* lets you do so.

CPU *primary_cpu*
Specifies the primary CPU on which the Reader process runs. The default is the CPU on which Manager runs.

BACKUPCPU *cpu*
Specifies an alternate CPU on which the process runs if the primary CPU becomes unavailable.

PRI *priority*
The NonStop priority for the process. This defaults to the priority assigned to the TACL process underlying the ADD.

PROGRAM *program_name*
This specifies the name of the program that Manager uses when starting the Reader process. Typically this is not entered, and Manager uses the HOST parameter in the GLOBALS files to determine the Oracle GoldenGate installation subvolume as indicated in the GGSSUBVOL option.

Example

```
The following example identifies a Reader process to monitor Oracle GoldenGate trail
AA on \NY.
READER EXTTRAIL \NY.$DATA5.GGSDAT.AA, PROCESS $GGRD1, CPU 1, PRI 180
```

READTHRULOCKS | NOREADTHRULOCKS

Valid for

Extract

Description

Use READTHRULOCKS to ignore record locks when performing a FETCHCOMPS or NOFETCHCOMPS operation. The default is to wait for locks to clear before reading the record. In some applications, this can help avoid a deadlock.

NOREADTHRULOCKS waits for locks to clear before reading records.

Default

```
NOREADTHRULOCKS
```

Syntax

```
READTHRULOCKS | NOREADTHRULOCKS
```

RECEIVEQWARN

Valid for

Logger

Description

Use RECEIVEQWARN to specify a maximum number of messages to hold in queue. Oracle GoldenGate issues an EMS warning when the maximum is exceeded.

Default

200

Syntax

```
RECEIVEQWARN num_messages
```

num_messages
The message threshold for the queue.

RENAMEBUMPDELAY

Valid for

CHGNOTE

Description

Use `RENAMEBUMPDELAY` to specify the number of seconds to bump the last modified timestamp on the `GGSCPU`*`nn`* files. By default, the last modified timestamp is changed to the current system time plus one second. If a user program that is bound with the Oracle GoldenGate Intercept library notes that `GGSCPU`*`nn`* has a new timestamp, it reevaluates file opens to make sure it has the proper file name. It continues reevaluating file opens until it reaches a time in the future (current time plus the bump specified with `RENAMEBUMPDELAY`).

You can increase the number of bump seconds into the future to address situations where the user application reevaluates file opens before an actual file rename is completed. Increasing the value will cause the user application to reevaluate file opens for a longer period. The default is the recommended setting.

Default

1

Syntax

```
RENAMEBUMPDELAY seconds
```

seconds
Can be any value from 1 through 15.

Example

```
RENAMEBUMPDELAY 2
```

REPERROR

Valid for

Replicat

Description

Use `REPERROR` to determine how Replicat responds to record errors that occur during replication. Using `REPERROR`, you can handle most record errors in a default manner and specific errors differently. For example, you can ignore duplicate record errors but abort processing in all other cases.

The parameter "HANDLECOLLISIONS | NOHANDLECOLLISIONS" describes how to handle missing and duplicate record errors as special cases.

See the Handling Replicat Errors for information on error handling for primary key updates, where multi-staged I/O can result in partial replication for non-audited files.

Default

```
ABEND
```

Syntax

```
REPERROR (error, response) | RESET
```

The following are possible *response* values and options:

```
REPERROR (error, [IGNORE | EXCEPTION | DISCARD | ABEND])

REPERROR (error, RETRYOP
[, MAXRETRIES number]
[, DELAYSECS seconds]
[, DELAYCSECS csecs])

REPERROR (error, TRANSABORT
[, MAXRETRIES number]
[, DELAYSECS seconds]
[, DELAYCSECS csecs])

REPERROR (error, FILEOP
[, MAXRETRIES number]
[, DELAYCSECS csecs]
[, DELAYSECS seconds]
[, ABEND | WARN | IGNORE | PURGETARGET]
[, CREATE | ALTER | RENAME | PURGE | SETMODE | CONTROL | CHANGELABEL])

REPERROR (error, RETRYOPEN
[, MAXRETRIES number]
[, DELAYSECS seconds]
[, DELAYCSECS csecs])

REPERROR (error [IOTYPE [,INSERT] [,UPDATE] [,DELETE]]
{IGNORE | EXCEPTION | DISCARD | ABEND | TRANSABORT})
```

error
Valid values are as follows.

DEFAULT
Sets a global response to all errors except those for which explicit REPERROR statements are specified.

DEFAULT2
Signals a backup default exception handling action when DEFAULT is set to EXCEPTION. Use DEFAULT2 when an exception map is not specified for a map that experiences an error.

error_number
A Guardian or SQL error number.

response
Can be one of the following:

ABEND
Roll back the transaction and terminate processing abnormally. ABEND is the default.

DISCARD
Log the error in the discard file but continue processing the transaction and subsequent transactions.

EXCEPTION
Handle the error as an exception. In anticipation of possible errors, you can create a separate `MAP` statement that executes only after an error. Use this `MAP` statement, for example, to map columns from a failed update statement into a "missing update" table. In the parameter file, specify the map that will handle the exception after the map that failed.

FILEOP
Specifies error handling behavior for each I/O type. The default is to `ABEND`. You cannot specify multiple actions for a given error and operation type, nor can you specify multiple operation types on a single statement.

> **Note:**
>
> `PURGETARGET` is valid only for the `CREATE` option on Enscribe files. It purges the target after a file create and retries the `CREATE`.

IGNORE
Ignore the error.

IOTYPE
Specifies behavior that should be applied for the designated types of operations. If `IOTYPE` is used, at least one of the operation types (insert, update, or delete) must be included.

RETRYOPEN
Retry a file open error. If the error is on file open or during the mapping evaluation, `RETRYOPEN` retries the operation indefinitely every 60 seconds. For example, if a file does not yet exist on the target platform, an operator could create it after seeing the open error, and processing will continue uninterrupted.

RETRYOP
Retry an insert, update, or delete operation. Use the `MAXRETRIES` option to limit the number of retries to the value of *`number`* For example, if Error 45 (file is full) indicates the table is out of extents, `RETRYOP` with `MAXRETRIES` gives you time to add extents so the transaction does not fail. Replicat abends after the specified number of `MAXRETRIES`.
To specify the length of time between attempts, set `RETRYDELAY` as described on "RETRYDELAY".

TRANSABORT
Abort the transaction and reposition to the beginning of the transaction. This will continue either until the record(s) are processed successfully or `MAXRETRIES` expires. If `MAXRETRIES` is not set, then the `TRANSABORT` action will loop continuously. To force an infinite loop of retries you can set `MAXRETRIES` to -1. Use the `DELAY` option to delay the retry.
The default delay between retries is 60 seconds, but this can be changed using the `DELAY` option.

The TRANSABORT option is useful for handling timeouts and deadlocks on databases that support those conditions.

RESET
Use RESET to remove all of the REPERROR settings made at the root level of the parameter file above the RESET.

> **Note:**
>
> The RESET option clears only the global error settings. It does not apply to and is not valid with REPERROR used within a MAP statement.

Examples

Example 1
This example shows valid actions and operation types for the option FILEOP.

```
REPERROR (11, FILEOP, RENAME, WARN)
REPERROR (14, FILEOP, RENAME, ABEND)
REPERROR (48, FILEOP, RENAME, ABEND)

REPERROR (10, FILEOP, CREATE, ABEND)
REPERROR (10, FILEOP, CREATE, PURGETARGET)
REPERROR (14, FILEOP, CREATE, ABEND)

REPERROR (11, FILEOP, PURGE, ABEND)
REPERROR (14, FILEOP, PURGE, ABEND)
REPERROR (48, FILEOP, PURGE, ABEND)

REPERROR (48, FILEOP, SETMODE,IGNORE)
REPERROR (48, FILEOP, CONTROL, IGNORE)
REPERROR (48, FILEOP, CHANGELABEL, ABEND)
```

Example 2
This example aborts processing for most file errors, but ignores duplicate record errors for both SQL tables (error -8227) and Enscribe files (Error 10).

```
REPERROR (DEFAULT, ABEND)
REPERROR (-8227, IGNORE)
REPERROR (10, IGNORE)
REPERROR (11, RETRYOPEN)
REPERROR (45, RETRYOP, MAXRETRIES 50)
```

Example 3
This example invokes an exception map for all errors on the ACCOUNT table. Errors on other tables cause Replicat to abend (due to the absence of an exception map for those tables).

```
REPERROR (DEFAULT, EXCEPTION)
REPERROR (DEFAULT2, ABEND)

-- The following MAP has no exception handler, so errors cause Replicat to
-- abend.
MAP $DATA1.DAT.PRODUCT, TARGET PRODUCT;

-- The following MAP has an exception handler, so errors will
```

```
-- cause Replicat to insert problem information into the target table.
MAP $DATA1.DAT.ACCOUNT, TARGET ACCOUNT;
INSERTALLRECORDS
MAP $DATA1.DAT.ACCOUNT, TARGET ACCOUNT_EXCEPTION,
        EXCEPTIONSONLY,
        COLMAP (ACCOUNT_NO = ACCOUNT_NO,
        OPTYPE =   @GETENV ("LASTERR", "OPTYPE"),
        DBERR =    @GETENV ("LASTERR", "DBERRNUM"),
        DBERRMSG = @GETENV ("LASTERR", "DBERRMSG"));
```

Example 4
This example discards inserts if an error 10 is encountered.

```
REPERROR (10, IOTYPE, INSERT, DISCARD)
```

This example ignores record not found (error 11) for updates and deletes.

```
REPERROR (11, IOTYPE, UPDATE, DELETE, IGNORE)
```

REPLACEBADCHAR

Valid for

Extract, Replicat

Description

Use `REPLACEBADCHAR` to substitute a specified value whenever an unprintable character is encountered during mapping or ASCII formatting.

`REPLACEBADCHAR` applies globally. If data is ASCII formatted and transported to a platform other than NonStop, you must specify the –r parameter in the Collector startup command.

Default

No replacement when mapping data; Replace with spaces when formatting ASCII data

Syntax

```
REPLACEBADCHAR char
[, SPACE]
[, NULL]
[, NONE]
[, UNPRINTABLE]
```

char
A character value with which to replace unprintable character bytes.

SPACE
Replaces unprintable character values with spaces.

NULL
Replaces the field with `NULL` whenever possible, otherwise, replaces the field with spaces.

NONE
Suppresses transformation of double-byte character set values to default characters when the incoming data is converted to ASCII format. This is useful when delivering double-byte character sets.

UNPRINTABLE

- When mapping data, rejects character fields or columns with unprintable data.
- When formatting data in ASCII or SQL output, outputs the word "UNPRINTABLE" or a question mark output for the field or column.

REPLACEBADNUM

Valid for

Extract, Replicat

Description

Use REPLACEBADNUM to specify a value to substitute whenever a numeric field that contains non-numeric data is encountered during mapping or ASCII formatting. This is most often seen in Enscribe data when, for example, a PIC 9(4) field contains spaces.

REPLACEBADNUM applies globally. If data is ASCII formatted and transported to a platform other than NonStop, you need to specify the -R parameter in the Collector startup command.

Default

Replace invalid numbers with zero when both mapping and formatting data

Syntax

```
REPLACEBADNUM number [, NULL] [, UNPRINTABLE]
```

number
Replaces the invalid field with a specified value.

NULL
Replaces the field with NULL whenever possible; otherwise, replaces the field with zero.

UNPRINTABLE

- When mapping data, rejects invalid numeric fields.
- When formatting invalid data in ASCII or SQL output, outputs a zero instead.

REPLICAT

Valid for

Replicat

Description

Use `REPLICAT` to link the current run with a Replicat group for continuous processing of trails. The group's checkpoints ensure that each record is processed exactly once.

To use `REPLICAT` establish the Replicat group with the GGSCI `ADD REPLICAT` command.

Syntax

```
REPLICAT group_name
```

group_name
The Replicat group name.

Example

```
REPLICAT FINANCE
```

REPNEWCOLUMNS | NOREPNEWCOLUMNS

Valid for

Replicat

Description

Use `REPNEWCOLUMNS` to tell Replicat to deliver SQL `ALTER TABLE ADD COLUMN` statements from the source database to the target database. `REPNEWCOLUMNS` can be applied on a table-by-table basis.

Default

```
NOREPNEWCOLUMNS
```

Syntax

```
REPNEWCOLUMNS | NOREPNEWCOLUMNS
```

Example

The following example delivers new columns to `TABLE1`, `TABLE2` and `TABLE4`, but not `TABLE3`.

```
REPNEWCOLUMNS
MAP \LA.$DATA1.DAT.TABLE1, TARGET \NY.$DATA1.DAT.TABLE1;
MAP \LA.$DATA1.DAT.TABLE2, TARGET \NY.$DATA1.DAT.TABLE2;
NOREPNEWCOLUMNS
MAP \LA.$DATA1.DAT.TABLE3, TARGET \NY.$DATA1.DAT.TABLE3;
REPNEWCOLUMNS
MAP \LA.$DATA1.DAT.TABLE4, TARGET \NY.$DATA1.DAT.TABLE4;
```

REPORT

Valid for

Extract, Replicat

Description

Use `REPORT` to specify times when a report is written to the report file. The report details the number of records extracted per file and table.

Default

`END`

Syntax

```
REPORT
[
AT hh:mm |
ON day_of_week |
AT hh:mm ON day_of_week |
END
]
```

`AT hh:mm`
The time of day. Using `AT` without `ON` generates a report at the specified time every day.

`ON day_of_week`
The day of the week. Valid values are `SUNDAY` through `SATURDAY`.

`END`
The end of the run.

Example

The following example generates a report every day at 5:00 pm and every Sunday at 1:00 am.

```
REPORT AT 17:00
REPORT ON SUNDAY AT 1:00
```

REPORTCOUNT

Valid for

Extract, Replicat

Description

Use `REPORTCOUNT` to write a count of records processed since the beginning of processing. Record counts can be output at scheduled intervals or after a specified number of records. Only the last `REPORTCOUNT` entry in the parameter file is used.

Syntax

```
REPORTCOUNT [EVERY]
count {RECORDS | SECONDS | MINUTES | HOURS}
```

`[EVERY]`
Specifies that a frequency follows.

count
The interval after which to write the count.

RECORDS | SECONDS | MINUTES | HOURS
Bases the interval on either the number of records or a time schedule.

Examples

Example 1

```
REPORTCOUNT EVERY 5000 RECORDS
```

Example 2

```
REPORTCOUNT EVERY 10 MINUTES
```

REPORTFILEEXTENTS

Valid for

GLOBALS, Extract, Replicat, Syncfile

Description

Use REPORTFILEEXTENTS to set the primary, secondary, and maximum file extents for the report file. This can be set at the global level, or in an Extract or Replicat parameter file. If you do not set it in the GLOBALS parameter file, your settings do not take effect until the second report file opens. The first report will contain default values since the report file is opened before an Extract or Replicat parameter executes.

Default

Primary is 32; Secondary is 32; Maximum is 900

Syntax

```
REPORTFILEEXTENTS (primary, secondary, maximum)
```

REPORTROLLOVER

Valid for

GLOBALS, Replicat, Syncfile

Description

Use REPORTROLLOVER to specify times at which the current report file is aged and a new one is created. Old report files are renamed *group_name*0, *group_name*1, etc.

Syntax

```
REPORTROLLOVER
{AT time |
AT time ON day_of_week}
```

AT *time*
The time of day. If AT is specified without ON, a new report file is created at the given time every day.

ON *day_of_week*
SUNDAY through SATURDAY.

REPORTTMFEXCEPTIONS

Valid for

Replicat

Description

Use REPORTTMFEXCEPTIONS for debugging purposes, generally in conjunction with Oracle GoldenGate Technical Support. REPORTTMFEXCEPTIONS reports when Replicat detects a TMFEXCEPTIONS type of condition. Messages are written to the report file in the following format:

```
location_id FILE target_file_name POS (seqno, rba)
```

Values for *location_id* include:

"track duplicate"
A duplicate record that occurred on insert was placed on the list of records to be tracked.

"backout of duplicate"
A TMF backout delete record was matched to a duplicate insert (so no processing is required and the duplicate error is no longer tracked).

"fix reversed insert"
A delete record matched to out-of-order duplicate insert

"update backout"
An update record matched to correct a duplicate record

"reapply update"
Attempting to process record from missing update list

"reapply duplicate"
Attempting to process record from duplicate list

"keep missing update"
A missing update/delete record was placed on list

"reversed update"
An insert record was overlaid with an update/delete from the missing updates list

"reversed insert add col"
When applying an update record to a record on the duplicate list, a column in the source update was not present in the record on the duplicate list. This column was added to the target record.

"reversed insert from update"
An update record was overlaid with a record on the duplicate list.

"reversed insert from delete"
A delete record was matched to an out-of-order duplicate insert

"Reapply duplicate rec failed"
The attempt to process the updated duplicate record failed.

"REVERSEWINDOWSECS expired for duplicate"
The timer expired before the reversed situation was corrected.

"REVERSEWINDOWSECS expired for missing delete"
The timer expired before the reversed situation was corrected.

"REVERSEWINDOWSECS expired for missing update"
The timer expired before the reversed situation was corrected.

Default

`OFF`

Syntax

```
REPORTTMFEXCEPTIONS {OFF | ON}
```

REPSQLLOG

Valid for

Replicat

Description

Use `REPSQLLOG` to redirect the log to a file other than the Replicat report file. The default is to write the SQL log to the report file.

If you specify the `SHOWSYNTAX` option of the `MAP` parameter, the text of the SQL statement is displayed before the statement is executed.

You can also redirect the SQL log with the define `=GGS_REPSQLLOG`.

Default

`APPEND`

Syntax

```
REPSQLLOG file_name [APPEND | PURGE | !]
```

file_name
The fully qualified path and file name of the alternative file, such as `$data1.sqllog.jsr01`.The *`file_name`* cannot be a structured file or SQL table, it can be a spooler, a process, a terminal or an `EDIT` file.

`APPEND`
The default. Appends current log information to an existing file.

PURGE
Purges any existing file at startup.

Example

```
REPSQLLOG $data1.sqllog.jsr01
```

RESTARTCOLLISIONS | NORESTARTCOLLISIONS

Valid for

Replicat

Description

Use `RESTARTCOLLISIONS` to enable `HANDLECOLLISIONS` until the first checkpoint is finished. After the first checkpoint is finished `RESTARTCOLLISIONS` is turned off.

Default

```
NORESTARTCOLLISIONS
```

Syntax

```
RESTARTCOLLISIONS | NORESTARTCOLLISIONS
```

RESTARTINTERVAL

Valid for

Manager

Description

Use `RESTARTINTERVAL` to reinitialize the restart count specified by the `MAXABENDRESTARTS` parameter.

When you specify `MAXABENDRESTARTS` *`count`*, the value of *`count`* is reset by default every 20 minutes if the process continues to run without abending. Specify `RESTARTINTERVAL` if you want to reinitialize at a different interval.

If `AUTORESTART` with `RESETMINUTES` has been set, `RESTARTINTERVAL` will be ignored.

For more information on using `MAXABENDRESTARTS` for restarting processes, see "MAXABENDRESTARTS".

Default

20

Syntax

```
RESTARTINTERVAL minutes
```

minutes
The number of minutes a restarted process must run successfully before the restart count specified by the `MAXABENDRESTARTS` parameter is reset.

RESTORE | NORESTORE

Valid for

Extract

Description

Use `RESTORE` to reload audit dumps on tape when the dumps are not available on disk. Use `NORESTORE` to avoid reloading tapes.

Default

`RESTORE`

Syntax

```
RESTORE | NORESTORE
```

RETRYDELAY

Valid for

Replicat

Description

Use `RETRYDELAY` to specify the delay between attempts at retrying a failed insert, update, or delete operation. Set the retry attempts with `REPERROR RETRYOP`, described under `REPERROR` on "REPERROR".

Syntax

```
RETRYDELAY seconds
```

seconds
The number of seconds between retry attempts

RETRYERR

Valid for

Extract

Description

Use `RETRYERR` to retry errors encountered while outputting records. For example:

- Extracted data is output to another node, and the network experiences a temporary problem. Rather than ending abnormally, you could specify a periodic retry of specific network-related errors numbers (such as 250).
- A file in the trail fills up prematurely (error 45). Messages indicating the problem alert an operator to increase `MAXEXTENTS` on the file, after which processing would resume automatically.

Syntax

```
RETRYERR {error_num | DEFAULT | EXPAND}
[, MAXRETRIES retries]
[, DELAY seconds]
```

error_num
Retries a particular error number.

DEFAULT
Retries Guardian errors not specifically dealt with in other RETRYERR entries.

EXPAND
Retries EXPAND-related errors. This does not cover TCP/IP-related problems.

MAXRETRIES *retries*
The number of retries before the process ends abnormally (default is 10). Set *retries* to zero to turn off retries for specific errors when a DEFAULT has been specified.

DELAY *seconds*
The number of seconds before the next retry (default is 20).

Example

The following example retries Error 45 every 30 seconds a maximum of 10 times, and retries Expand errors every minute for an hour.

```
RETRYERR 45, DELAY 30, MAXRETRIES 10
RETRYERR EXPAND, DELAY 60, MAXRETRIES 60
```

RMTBATCH

Valid for

Extract

Description

Use RMTBATCH to specify the name of a remote target file to which Extract writes extracted records. RMTBATCH is like RMTFILE in that it specifies the name of a remote target file. However, rather than outputting the data in a Oracle GoldenGate proprietary format, Extract outputs the data to a batch file on the target system. The batch file name is determined by a combination of the target file name and the source file name.

Using this feature, you can 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.

Use wildcards to accommodate different file names with a single RMTBATCH parameter entry.

RMTBATCH enables full restart capabilities without loss or repetition of data. This is enabled using Extract checkpoints (GGSCI ADD EXTRACT command).

When using direct read functions to transfer batch files with RMTBATCH, use the following Extract parameters whenever the record length is variable and the source file type is entry-sequenced:

- SYSKEYCONVERT USEBYTEPOSITION
- POSITIONFIRSTRECORD

With these parameters, Extract can build the target file in all cases, including after a recovery.

If SYSKEYCONVERT is specified, you cannot mix RMTBATCH activity with standard extraction of data for replication.

To create OS/390 QSAM files, you can specify certain file creation parameters in a configuration file on the OS/390 platform or using RMTBATCH.

RMTBATCH has the following dependencies:

- You must include at least one EXTFILE, EXTTRAIL, RMTFILE, RMTTRAIL, or RMTBATCH entry in your parameter file.
- 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.

Syntax

```
RMTBATCH destination_file_spec
[, PARAMS param [, param, ...]];
```

> **Note:**
>
> See also the information on RMTBATCH for QSAM on "RMTBATCH for QSAM".

destination_file_spec
Either a file name, or a file name combined with wildcard characters. The wildcard characters are * and %. The name of the source file is substituted for each occurrence of * in the file specification, and the name of the source subvolume is substituted for each occurrence of %.

PARAMS *param*
param can be any one of the following:

AUTOTRUNCATE
Executed by Extract or Collector, depending on its placement on the command line.
If AUTOTRUNCATE is specified before the parameters are defined, Extract deletes the contents of the batch file when it positions to the first record. When you periodically move the same file to the target system, use this parameter to delete previous data from the file.
If AUTOTRUNCATE is specified after the parameters are defined, Collector deletes the contents of the batch file when it positions to the first record.

BLKSIZE *block_size*
Indicates the block size of the source file (default is 4096). This option is required when the source file is entry-sequenced and its block size is less than 4096.

CLOSEWIN *minutes*
Closes the target file after *minutes* of inactivity.

CLOSEACTION "*system_command*"
Submits a shell script or system command to the system upon file close, when close was a result of completed processing of the corresponding source file (indicated by a "file close on source" message). Extract waits for the command to complete. To return control to Extract immediately, specify a `nowait` type operation in the script or command (as with the `&` in UNIX).

FTYPE {E | K | R | U}
Identifies the source file type to the Collector (`E` is entry-sequenced, `K` is key-sequenced, `R` is relative, `U` is unstructured). The default is `E`, entry-sequenced. The Collector uses `FTYPE` to compute an appropriate record position into the target file (which is always unstructured).
Because keys are identified differently for different file types, this parameter is required if the file type is relative or unstructured.

RECLEN *record_length*
Identifies the length in bytes of each record in the target file. Use this to pad records that are shorter than *record_length* to a constant length. Records are padded with spaces.

> **Note:**
>
> All `RMTBATCH` statements must end with a semicolon (;).

Example

Assume the following configuration:

```
RMTHOST ggs2, MGRPORT 7809
RMTBATCH /usr/ggsdat/%_*,
PARAMS FTYPE E, BLKSIZE 2048, RECLEN 1916;
FILE $DATA2.GGS.TL*;
```

Also assume that a new `TL` file is created every day, with the naming convention of `TLyymmdd`.

On April 5, 2010, the file `$DATA2.GGS.TL100405` is created. On April 6, `$DATA2.GGS.TL100406` is created. On the target system, the following files are created, each containing data from the corresponding source file:

```
/usr/ggsdat/GGS_TL100405
/usr/ggsdat/GGS_TL100406
```

`FTYPE E` identifies each source file as entry-sequenced; `BLKSIZE` specifies the block length of those files as `2048`; and `RECLEN 1916` outputs all records as `1916` bytes by padding short records with spaces.

RMTBATCH for QSAM

The following parameters are for QSAM files only:

TARGETFTYPE {QSAMV | QSAMF}
Mandatory parameter for QSAM files. Specify QSAMV for variable-length MVS QSAM files or QSAMF for fixed-length. MVS QSAM files.

TARGETRECLEN *record_length*
Mandatory parameter for QSAM files. Specify the length in bytes of a fixed-length record or the maximum length in bytes for variable-length records.`record_length`can be from `1` to `32760`.

TARGETBLKSIZE *block_length*
Parameter for QSAM files. Specify a multiple of the record length for fixed-length records, or the maximum record length +4 for variable-length records.`block_length`can be from `1` to `32760`.

VOLUME *serial_number* [, *serial_number*, ...]
Mandatory parameter for QSAM files. Identifies the serial numbers of the volumes on which the target data set will reside. Specify up to `255` volume serial numbers.

SPACE (*primary_quantity*, *secondary_quantity*, *unit*)
Mandatory parameter for QSAM files. Requests space for a new data set.

primary_quantity
The number of tracks, cylinders, or blocks to be allocated for a new data set.

secondary_quantity
The number of additional units to allocate if more space is needed.

unit
The units in which space is requested: `TRK` for tracks, `CYL` for cylinders, `BLK` for blocks.

For examples of QSAM parameters, see "QSAM Configuration Parameters".

RMTBATCH and Generation Data Groups

Use `RMTBATCH` to create and/or access IBM QSAM Generation data sets within Generation Data Groups (GDG). This feature catalogs successive updates, or generations, of data sets within a GDG. It is typically used on BASE24 Extract and Super Extract output files, but can be used on any entry-sequenced file type.

Generation data sets have sequentially ordered absolute and relative names that represent their age. (Smaller absolute names indicate older data.) A relative name is a signed integer indicating the latest generation (0), next to latest generation (-1), etc. The relative number +1 creates a new generation each time it is used.

Use the following parameters with `RMTBATCH` to create or access your generation data sets.

LIKE
Copies the allocation attributes of a model data set to specify the allocation attributes of a new data set. The model data set must be an existing, cataloged data set.

UNIT
Asks the systems to place the data set on a specific device, device type, or group of devices as another data set.

DCBSN (-DCB Model data set)
Names a catalogued data set. Instructs the system to copy data control block information from the data set's label.

To create a generation data set using RMTBATCH:

1. Create a Generation Data Group index using IDCAMS.
2. Create an Extract group and parameter file.
3. Create a RMTBATCH parameter file as follows:
   - Specify your GDG file name then a relative number for the GDG file (negative integer, positive integer, or zero)
   - Add the -P parameter
   - Add an Extract as a file type entry.

Example

Sample Extract Parameter File with RMTBATCH. The remote batch statement is in **bold**.

```
EXTRACT EXTTLF1
DISCARDFILE $DATA1.DISCARD.EXTFLT1
SYSKEYCONVERT USEBYTEPOSITION
ALTINPUT OPENTIMEOUT 5, TEMPLATE $DATA01.EXTRACT.P*, USENEXTMODIFIED
POSITIONFIRSTRECORD
RMTHOST OS390, MGRPORT 7809, params "-c ON -P qsamvb.prm"
RMTBATCH file.acct(+1),params ftype u, closewin 2;
File $DATA01.EXTRACT.P*;
```

RMTFILE

Valid for

Extract

Description

Use RMTFILE when the remote Oracle GoldenGate trail is a flat file.

All FILE and TABLE entries after the current entry but before the next trail parameter (EXTFILE, EXTTRAIL, RMTBATCH, RMTFILE, RMTTRAIL) result in output to the current trail.

Remote Oracle GoldenGate trails are used over TCP/IP connections only. Do not specify an Expand node name in the *file_name*, even if the remote system is also connected with Expand. To specify a trail on a different NonStop node over an Expand connection, use the EXTFILE or EXTTRAIL parameters.

RMTFILE must be preceded by a RMTHOST statement, and it must precede any TABLE, FILE or MAP statements.

You must include at least one EXTFILE, EXTTRAIL, RMTBATCH, RMTFILE or RMTTRAIL entry in your parameter file. It is required for initial loads that use a trail.

Syntax

```
RMTFILE file_name
[, PURGE | APPEND]
[, EXTENTS (primary, secondary, maximum)]
```

```
[, MAXFILES num_files]
[, MEGABYTES megabytes]
[, OWNER group_number, user_number]
[, SECURE "[, FORMAT RELEASE major.minor] rwep"]
```

FORMAT RELEASE major.minor

Specifies the trail format of the data that is sent by Extract to a trail, file, or (if a remote task). The value tells the reader process whether the data records are of a version that it supports. The format depends on the version of the Oracle GoldenGate process. Newer Oracle GoldenGate versions contain different characteristic than older ones.

- FORMAT is a required keyword.
- RELEASE specifies an Oracle GoldenGate release version. major is the major version number, and minor is the minor version number. The X.x must reflect a current or earlier, generally available (GA) release of Oracle GoldenGate. Valid values are 9.0 through the current Oracle GoldenGate X.x version number, for example 11.2 or 12.1. (If you use an Oracle GoldenGate version that is earlier than 9.0, specify either 9.0 or 9.5.). The release version is programmatically mapped back to an appropriate internal compatibility level. The default is the current version of the process that writes to this trail.

file_name
The remote file on the remote system.

PURGE | APPEND
Purges *file_name* before capturing data, or appends the extracted data to *file_name*.

EXTENTS (*primary*, *secondary*, *maximum*)
Defines the primary, secondary and maximum extents when the target is a NonStop system.

MAXFILES *num_files*
Enables a sequence of files to be created on the target system, rather than a single file. When MAXFILES is specified, a six-digit sequence number is appended to the end of *file_name* for each file. MAXFILES does not actually limit the number of files created. When using MAXFILES, MEGABYTES should also be specified in order to explicitly set the maximum size of each file in the sequence.

MEGABYTES *megabytes*
Sets up the maximum size of the file, or sequence of files if you specified MAXFILES. The maximum size for each file is 2 gigabytes.

OWNER *group_number*, *user_number*
Determines which user group owns the RMTFILE.

SECURE "*rwep*"
Sets the standard Guardian security for read, write, execute, and purge operations on the RMTFILE.

Example

The following example extracts data from SALES to a UNIX or Windows system, depending on region. Collectors are started dynamically by Manager. The Manager port for NY is the default, port 7809. To resolve any possible conflict, the LA port number is explicitly identified. Note that the TCPIPPROCESSNAME is also specified (the default is $ZTC0).

```
TCPIPPROCESSNAME $ZTC4
RMTHOST NY, MGRPORT 7809
RMTFILE /usr/dat/sales1
TABLE $DATA5.SALES.ORDERS WHERE (REGION = "EAST");
TABLE $DATA5.ACCTING.RECEIPTS WHERE (REG = "E");
RMTHOST LA, MGRPORT 7888
RMTFILE d:\dat\wstsales
TABLE $DATA5.SALES.ORDERS WHERE (REGION = "WEST");
TABLE $DATA5.ACCTING.RECEIPTS WHERE (REG = "W");
```

RMTHOST

Valid for

Extract

Description

Use `RMTHOST` to identify the target system and TCP/IP port number for subsequent `RMTBATCH`, `RMTFILE`, or `RMTTRAIL` entries. You can make one or more `RMTHOST` entries to identify multiple target nodes. Only one entry is active at any time.

Specifying the Process Name

There are multiple ways to specify the `TCPIPPROCESSNAME` when the NonStop TCP/IP process name is not `$ZTC0`. When assigning a process name, the system selects the option to use based on this hierarchy:

1. `RMTHOST TCPIPPROCESSNAME` option in the Extract parameter file
2. Stand-alone parameter `TCPIPPROCESSNAME` entry in the Extract parameter file before the `RMTHOST` entry.
3. `DEFINE =TCPIP^PROCESS^NAME` added before the Extract process was started
4. `GLOBALS` entry for `TCPIPPROCESSNAME`
5. `MGR`'s entry for `TCPIPPROCESSNAME` when process is started from GGSCI
6. Default `$ZTC0`

Using Streaming

By default Extract uses `STREAMING` and does not wait for a response from the remote host Collector before sending the next message. With `STREAMING`, Extract waits for a response only when it needs to checkpoint. It can potentially send multiple megabytes of data without waiting and therefore improve through-put.

Extract checks the remote Collector when using `STREAMING`. If the Collector does not support streaming, Extract reverts to waiting for a response before sending the next message. To always wait for a response before sending, specify `NOSTREAMING`.

> **Note:**
>
> `RMTTASK` does not allow `STREAMING`. If the `RMTTASK` parameter is used with the `STREAMING` option, `STREAMING` is disabled.

Identifying Static and Dynamic Collectors

RMTHOST can be used to identify a dynamic Collector that is started by the Manager or a static Collector that is defined and started by the user.

When using the static Collector method, the target computer must be running the Collector program before starting Extract.

When using the dynamic collector method, Extract requests Manager on the remote system to start a Collector process. In this case, Collector terminates itself when the connection is lost or Extract terminates. If network problems occur, Extract and Manager coordinate restart and retry efforts.

Syntax

```
RMTHOST {host_name | ip_address}
[, PORT port_number | MGRPORT port_number]
[, IPINTERFACE ip_address]
[, STREAMING | NOSTREAMING]
[, TCPIPPROCESSNAME process_name [@ip_address]]
[, TIMEOUT seconds]
[, USEIPV4ONLY]
[, dynamic_options]
```

host_name
The remote host name. If *host_name* is specified, do not specify *ip_address*

ip_address
The IP address of the host. Either version (IPv4 or IPv6) of the address internet protocol is acceptable. If *ip_address* is specified, do not specify *host_name*.

PORT ***port_number***
An optional argument specifying the port for the static Collector. The default port is 7819.

MGRPORT ***port_number***
Used when Manager starts the dynamic Collector. Can be DEFAULT or the port number on which Manager is listening. The default port number is 7809. Do not specify a Collector port when Manager dynamically starts the Collector.

IPINTERFACE ***ip_address***
Restricts Extract to the specified IP address.

STREAMING | NOSTREAMING
Specifies whether Extract will wait for a response from the Collector before sending the next message.

> **STREAMING**
> Extract will wait for a response only when it needs to checkpoint, otherwise it will not wait before sending the next message. STREAMING is the default.
>
> **NOSTREAMING**
> Use NOSTREAMING to turn off STREAMING and cause Extract to wait for a response before sending the next message.

TCPIPPROCESSNAME *process_name @ip_address*
process_name restricts Extract to the specified process name. The optional *@ip_address* can be used instead of IPINTERFACE to force the process to bind to a specified IP address.

TIMEOUT *seconds*
Specifies the number of seconds that Server Collector or an Initial Load Replicat should wait for a connection to be made. The allowed values are 20 to 300 seconds. 300 seconds is the default.

USEIPV4ONLY
Forces the use of version 4 of the address internet protocol (IPv4) when both IPv4 and IPv6 protocols are configured and compatible.

dynamic_options
Specifies options for a dynamic Collector. Can be any of:

CPU *cpu_num*
The CPU in which Manager will start the Collector process when NonStop is the target system.

COMPRESS | NOCOMPRESS
Compresses outgoing blocks of extract records. The destination Collector decompresses the data stream before writing it to the remote file or remote trail. This typically results in compression ratios of at least 4:1 and sometimes much better. However, compression can require significant CPU resources.

COMPRESSTHRESHOLD
Sets the minimum block size for which compression is to occur. The default for compression is 1000 blocks.

ENCRYPT *type*
Encrypts the data stream sent to the remote host. The destination Collector decrypts the data stream before writing it to the remote file or remote trail. For details on the encryption algorithm, contact Oracle Golden Gate Support.
type is one of: NONE, GGS, or BLOWFISH. NONE specifies no encryption. GGS specifies a default key generated by Oracle GoldenGate. BLOWFISH specifies a user-defined key.
The destination Collector must be version 5.40 or above.

KEYNAME *keyname*
KEYNAME indicates the logical name of an encryption key to look up in the ENCYKEYS file. KEYNAME is required when ENCRYPT is set to BLOWFISH. The KEYNAME value is validated at startup of Extract or Collector, and if it is invalid, the process abends with an error message.

PARAMS "*collector_parameters*"
See the parameter description in "Collector Parameters". When the target system is NonStop, this parameter is rarely required.

PRI *priority*
The NonStop system priority at which Manager will start the collector process when the target is a NonStop system. Consult the HP NonStop documentation for more information about NonStop priorities.

NAME $*process*
Sends request to Manager to start a specific process as part of the RMTHOST startup request.

Examples

The following examples use different RMTHOST options to control the IP address.

Example 1

The following example sets an IP address using the TCPIPPROCESSNAME *process_name@* option.

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

Example 2

The following example sets an IP address using the IPINTERFACE option.

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

Example 3

The following example forces use of the IPv4 address when both versions are configured.

```
RMTHOST host01, MGRPORT 12345, USEIPV4ONLY
```

RMTHOSTALT

Valid for

Extract

Description

Use RMTHOSTALT to identify an alternative IP address and Collector in case the primary address or collector cannot be reached. After n retries of any TCP/IP errors (default is five), Extract attempts to connect and send data over the connection specified by RMTHOSTALT.

Specify RMTHOSTALT immediately after the corresponding RMTHOST entry. RMTHOSTALT applies only to the previous RMTHOST. Only one RMTHOSTALT entry is allowed per RMTHOST entry.

RMTHOSTALT requires a separate collector to be running that will accept data in the event that the RMTHOST becomes unavailable.

You can make one or more RMTHOST entries to identify multiple target nodes. Only one entry is active at any time.

In the event that a switch is made to the alternate address, and more errors occur, attempts will be made to switch back to the primary path.

RMTHOSTALT does not support dynamically created collectors. To use RMTHOSTALT, you must specify the RMTHOST PORT number explicitly (the static method).

Syntax

```
RMTHOSTALT {host_name | ip_address}, PORT port_number
[, IPINTERFACE ip_address]
[, TCPIPPROCESSNAME process_name [@ip_address]]
[, USEIPV4ONLY]
```

host_name
The host name. If *ip_address* is specified, do not specify *host_name*.

ip_address
The IP address. Either version (IPv4 or IPv6) of the address internet protocol is acceptable. If *host_name* is specified, do not specify *ip_address*.

port_number
The port number. There is no default port for RMTHOSTALT. The port entered here is required and must match the port on which the target Collector is listening.

IPINTERFACE *ip_address*
Restricts Extract to the specified IP address.

TCPIPPROCESSNAME *process_name @ip_address*
Restricts Extract to the specified process name. The optional @ip_addresss restricts Extract to the specified IP address.

USEIPV4ONLY
Forces the use of version 4 of the address internet protocol (IPv4) when both IPv4 and IPv6 protocols are configurated and compatible.

Example

The following example causes data to be output over TCP/IP process $ZTC0 to the system designated by 192.0.2.1 on port 7830. If errors are encountered, and three consecutive retries fail, data is output over TCP/IP process $ZTC1 to the system designated by 192.0.2.2 on port 7831.

```
TCPIPSWITCHERRS 3
RMTHOST 192.0.2.1, PORT 7830, TCPIPPROCESSNAME $ZTC0
RMTHOSTALT 192.0.2.2, PORT 7831, TCPIPPROCESSNAME $ZTC1
RMTTRAIL $DATA5.GGSDAT.RT
```

RMTTASK

Valid for

Extract

Description

Use RMTTASK to create a one-time processing task on the target system. RMTTASK directs the Manager process on the target system to start Replicat process automatically to load the data and then stop it when the task is finished. This parameter is used for initial data loads.

A RMTHOST statement must precede the RMTTASK statement. RMTTASK is required for batch tasks.

Syntax

```
RMTTASK task_name, GROUP group_name
[, PARAMS "runtime_[, params"]
```

task_name
Enter `REPLICAT`, currently the only task on the remote system.

`GROUP` ***group_name***
The name of the initial-load Replicat group on the target system. This option defines the target Replicat group processes the load data.

`PARAMS "`*runtime_params*`"`
Optional. Specifies run-time parameters to pass to the initial-load Replicat process other than standard parameters, such as report file, parameter file, and group name, which are supplied automatically.

Example

In the following example, records from table `PRODUCT` and `ACCOUNT` are sent to initial-load tasks on target system `NY`.

```
EXTRACT EXTTASK
RMTHOST NY, MGRPORT 7809
RMTTASK REPLICAT, GROUP tabload
TABLE $DATA.MASTER.PRODUCT;
TABLE $DATA.MASTER.ACCOUNT;
```

RMTTRAIL

Valid for

Extract

Description

Use `RMTTRAIL` to establish the current remote trail on a remote system connected by TCP/IP. All `FILE` and `TABLE` entries after the current entry but before the next Extract file parameter (`EXTFILE`, `EXTTRAIL`, `RMTBATCH`, `RMTFILE`, `RMTTRAIL`) result in output to the current trail.

The remote trail must correspond with a `RMTTRAIL` entry created with GGSCI. The Extract group in the parameter file must match the Extract entry linked with the `RMTTRAIL`. For more information about adding remote Oracle GoldenGate trails, refer to"ADD RMTTRAIL".

Syntax

```
RMTTRAIL file_[, FORMAT RELEASE major.minor] prefix
```

FORMAT RELEASE major.minor

Specifies the trail format of the data that is sent by Extract to a trail, file, or (if a remote task). The value tells the reader process whether the data records are of a version that it supports. The format depends on the version of the Oracle GoldenGate process. Newer Oracle GoldenGate versions contain different characteristic than older ones.

- FORMAT is a required keyword.
- RELEASE specifies an Oracle GoldenGate release version. major is the major version number, and minor is the minor version number. The X.x must reflect a current or earlier, generally available (GA) release of Oracle GoldenGate. Valid values are 9.0 through the current Oracle GoldenGate X.x version number, for example 11.2 or 12.1. (If you use an Oracle GoldenGate version that is earlier than 9.0, specify either 9.0 or 9.5.). The release version is programmatically mapped back to an appropriate internal compatibility level. The default is the current version of the process that writes to this trail.

file_prefix
The path name of the remote file. A six character sequence indicator is added to each file in the remote trail to make up individual file names.

Remote Oracle GoldenGate trails are used over TCP/IP connections only. Do not specify an Expand node name in the *file_prefix*, even if the remote system is also connected with Expand. To specify a trail on a different NonStop node over an Expand connection, use the EXTFILE or EXTTRAIL parameters.

Example

The following example assumes that an Extract group FINANCE was established with GGSCI. The example also assumes that two remote Oracle GoldenGate trails were created in GGSCI and associated with EXTRACT FINANCE. This configuration sends ACCOUNTS records to the XX trail, and ORDERS to the YY trail. The parameter file includes the following commands:

```
EXTRACT FINANCE
RMTHOST 192.0.2.2, MGRPORT 7809
RMTTRAIL /ggs/dirdat/XX
FILE $DATA2.FINANCE.ACCOUNTS;
RMTTRAIL /ggs/dirdat/YY
FILE $DATA3.FINANCE.ORDERS;
```

ROLLOVER

Valid for

Extract

Description

Use ROLLOVER to specify times at which local or remote trails automatically roll over, in sequence. Each ROLLOVER rule applies to all Oracle GoldenGate trails and remote trails.

Use ROLLOVER to extract data continuously but create files representing distinct periods of time (for example, each day). Also, to send extracted data to external systems periodically, use ROLLOVER to cut off one file and start the next.

Files roll over only between the output of distinct transactions, not in the middle of a transaction. Checkpoints are recorded as soon as files roll over to ensure the previous files are no longer required by Extract.

Rollover occurs if the rollover conditions are met during the run. In other words, if ROLLOVER ON TUESDAY is specified, and Extract starts on Tuesday, rollover will not occur until the next Tuesday (unless other ROLLOVER rules are specified).

You can specify up to thirty rollover rules.

Syntax

```
ROLLOVER
[AT hh:mm]
[ON day_of_week]
[AT hh:mm ON day_of_week]
[AT END]
[REPORT]
```

AT *hh:mm*
The time of day. Using AT without ON generates a report at the specified time every day.

ON *day_of_week*
The day of the week. Valid values are SUNDAY through SATURDAY.

AT END
A STOP requested from GGSCI or the end of a SPECIALRUN.

REPORT
A report regarding the number of records extracted from each table and file since the last report was generated. The report represents the number of records output to the corresponding file (unless other reports are generated by means of the REPORT parameter).

Example

The following example rolls over local and remote Oracle GoldenGate trails every day at 3:00 p.m. as well as every Sunday at 8:00 a.m. For instance, if the file $DATA2.EXTDATA.AA000122 is the current file Monday morning, then at 3:00 p.m. on Monday AA000122 will be closed and AA000123 will be created.

```
ROLLOVER AT 15:00
ROLLOVER AT 08:00 ON SUNDAY
```

SETENV

Valid for

Extract, Replicat

Description

Use SETENV to set a NonStop environment variable. When Extract or Replicat starts, it uses the specified value instead of the one set in the environment.

Use one SETENV statement per variable to be set. Any variables set in the SETENV statement override any existing variables set at the operating-system level.

Setting Variables

The results from setting variables with SETENV vary based on the quotation marks used.

- **No quotation marks**

  When no quotation marks are used, the result is upshifted so for SETENV (NAME = name1), the result is NAME1.

For example:

```
SETENV repl = replicat
Set environment variable (REPL=REPLICAT)
SETENV (repa = repall)
Set environment variable (REPA=REPALL)
?repl ?repa
2013-05-06 14:33:48 OGG ?REPL = REPLICAT
?REPA = REPALL.
```

- **Double quotation marks**

  When double quotation marks are used, the quotation marks are removed from the result, which is otherwise unchanged. For `SETENV NAME = "Name"`, the result is `Name`.

  For example:

```
SETENV name = "Name"
Set environment variable (NAME=Name)
MAP $data01.abdat.tcustmer, TARGET $data02.abdat.tcustmer,
 COLMAP (USEDEFAULTS, name = ?name);
2013-05-06 14:33:48 OGG ?NAME = Name.
```

- **Outer single quotation marks**

  When the value is enclosed in single quotation marks, the result is enclosed in double quotation marks and otherwise unchanged. For `SETENV NAME = 'a literal'`, the result is `"a literal"`.

  For example:

```
SETENV name = 'With Single quotes, we get a literal'
Set environment variable (NAME="With Single quotes, we get a literal")
MAP $data01.abdat.tcustmer, TARGET $data02.abdat.tcustmer,
 COLMAP (USEDEFAULTS, name = ?name);
2013-05-06 14:33:48 OGG ?NAME ="With Single quotes, we get a literal".
```

- **Outer double quotation marks**

  When outer double and inner single quotation marks are used, the result is still everything within the double quotation marks unchanged. For `SETENV NAME = "'name'"`, the result is `'name'`.

  For example:

```
SETENV WILL = "'WILL'"
Set environment variable (WILL='WILL')
MAP $data01.abdat.tcustmer, target $data02.abdat.tcustmer,
 COLMAP (USEDEFAULTS),
    SQLEXEC (ID ex1, TRACE ALL,  BEFPREFILTER, ERROR RAISE,
       QUERY " select * from $data01.tssout.tcustmer "
       " WHERE cust_code = ?WILL ",
       PARAMS (c1 = cust_code ) );
2013-05-06 14:33:48 OGG ?WILL = 'WILL' .
```

- **Embedded quotation marks**

  Use the escape character to retain embedded quotation marks. For `SETENV NAME = 'John\'s'`, the result is "John's".

  For example:

```
SETENV name ='Use the escape character for the embedded quote in John\'s name'
Set environment variable (NAME="Use the escape character for the embedded quote
in John's name")
```

```
MAP $data01.abdat.tcustmer, TARGET $data02.abdat.tcustmer,
 COLMAP (USEDEFAULTS, name = ?name);
2013-05-06 14:33:48 OGG ?NAME ="Use the escape character for the embedded quote
in John's name".
```

Default

None

Syntax

```
SETENV (environment_variable = value)
```

environment_variable
The name of the environment variable to be set.

'*value*'
A value for the specified variable. The value may be enclosed in quotes. See Setting Variables in the *Description* section for details on how single and double quotation marks are interpreted.

Example

```
EXTRACT GGS01
SETENV (TRAIL1=$DATA01.LOGGER.A1)
SETENV (TRAIL2=$DATA01.LOGGER.A2)
FILE $*.*.*;
EXTTRAIL ?TRAIL1
FILE $*.*.*
EXTTRAIL ?TRAIL2
```

SHORTREADDELAY

Valid for

Extract

Description

Use `SHORTREADDELAY` to optimize system resource usage when Extract runs online. `SHORTREADDELAY` instructs Extract to delay a specified number of seconds whenever a block shorter than the optimal block size is read from a trail. This parameter enables efficient processing of many records at once. By specifying a longer delay, data is sometimes not replicated as quickly.

Syntax

```
SHORTREADDELAY seconds
```

seconds
The number of seconds to delay.

SOURCEDEFS

Valid for

Extract, Replicat

Description

Use SOURCEDEFS to specify the name of the file on the target system that contains the source table and file definitions. Create the SOURCEDEFS file by running the DEFGEN utility on the source system. Once created, the file must be transferred to the target system before running Replicat.

You can use SOURCEDEFS to facilitate faster loading of definitions at startup rather than accessing remote SQL catalogs or DDL dictionaries for the information.

You must specify SOURCEDEFS before any MAP entries that require related source definitions. Multiple SOURCEDEFS entries can exist within the same parameter file if more than one file holds the necessary definitions (for example, if each SOURCEDEFS file held the definitions for a distinct application).

SOURCEDEFS is required only when the source system is unreachable through Expand.

Syntax

```
SOURCEDEFS file_name [OVERRIDE]
```

file_name
The name of the file containing the data definitions.

OVERRIDE
Use the OVERRIDE option in an Extract pump or Replicat to ignore metadata in the trail.

> **Note:**
>
> Consult Oracle Support before you use this option.

SOURCEISFILE | SOURCEISTABLE

Valid for

Extract

Description

Use SOURCEISFILE or SOURCEISTABLE to specify whether the source is a file or a table. In many cases, source and target files are synchronized with utilities such as FUP DUP or SQLCI LOAD. These utilities establish a point at which subsequent changes made to the source can be applied accurately at the target. However, if mapping is specified between source and target, or if the target is a non-NonStop database, conversion must take place before load.

SOURCEISFILE reads and processes records from the source file itself rather than the audit trail. If you specified user exits or column maps when Extract processes database changes through the audit trail, also specify SOURCEISFILE so that the same mapping can be applied before a full-scale file load. Therefore, only one set of mapping routines is required for managing both full synchronization and incremental change activities.

Syntax

```
SOURCEISFILE | SOURCEISTABLE
[, FASTUNLOAD | FASTUNLOADSHARED]
[, SQLPREDICATE]
[, SELECTVIEW]
[, RESTARTCHECKPOINTS]
```

FASTUNLOAD
Processes the file or table several times faster than the default method. Records are output 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.
`FASTUNLOAD` fails if other openers exist for the file or table (the file is opened in protected mode).

FASTUNLOADSHARED
Allows a shared open of the source file or table. Use this only on files that are not receiving updates at the same time data is being extracted.

SQLPREDICATE
Uses any `WHERE` clause designated as part of a `TABLE` entry as criteria when selecting data from the source table. `SQLPREDICATE` can dramatically reduce extraction time when only part of the source table needs to be extracted. This option only works when the source is an SQL table.
You can start separate Extract processes to retrieve data from selected partitions from large tables in parallel. Each process selects a range of data from one or more partitions. When `SQLPREDICATE` is specified, only relevant partitions are scanned.

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).
`SELECTVIEW` is required to process multiple table joins and aggregation views. Use `SELECTVIEW` to format complex query output for compatibility with other platforms. Rather than using standard utilities that output a record at a time, this method takes advantage of Extract's high performance buffering and cross-platform features.

RESTARTCHECKPOINTS
Instructs Extract to save checkpoint information allowing restart at the last record read if the process stops or abends. See"Using RESTARTCHECKPOINTS" for conditions and restrictions.

Using RESTARTCHECKPOINTS

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`.

`RESTARTCHECKPOINTS` is valid only for Extracts added from `GGSCI` using `SOURCEISFILE` or `SOURCEISTABLE`.

A positioned restart is valid only if RESTARTCHECKPOINTS is in the parameter file when the Extract starts initially. The process will start from the beginning of the file if RESTARTCHECKPOINTS is added or removed for any subsequent restart.

Either SOURCEDEFS or DICTIONARY and DEF are required for Enscribe source files.

You must *not* make any of the following changes to the parameter file when restarting from a position other than the beginning of the file. If you must make these changes, you must delete and re-add the Extract group.

- Removing checkpoint information by:
  - Removing the current Data File Checkpoint file from the parameter file. This file can be identified with the INFO EXTRACT command.

```
GGSCI> INFO EXTRACT EXTPART

Extract  EXTPART      Last Started 2010-12-06 12:39  Status RUNNING
Task type             SourceIsFile
Process  $TSE01       Checkpoint Lag: unknown
Data File Checkpoint  \NY.$DATA02.ACDATA.ACPART
.
.
.
```

  - Removing the current output checkpoint. This is also identified in the INFO EXTRACT command.

```
GGSCI> INFO EXTRACT EXTPART
.
.
.
                                        Record       30977
Target Extract Trails                                          Rba     Max MB
  \NY.$DATA02.ACDATA.TX000001                                2168775      3.
```

- Adding or removing the FASTUNLOAD option
- Adding a new EXTFILE, RMTFILE, or RMTTASK
- Changing the source file or table significantly, such as changing the key columns or their length or datatype
- Changing a STARTKEY or ENDKEY in a way that alters the file access method.

It is not recommended that you add files or tables to the parameter file after start-up and before a restart. They may not be extracted.

Replicat processes require the HANDLECOLLISIONS parameter during a restart of a RMTTASK Extract.

SPECIALRUN

Valid for

Extract, Replicat

Description

Use SPECIALRUN for one-time batch processing. SPECIALRUN tasks process activity that occurs within a time period, such as the changes logged between 8 a.m. and 3 p.m. on a particular date. Checkpoints are not recorded so the process can not be restarted

from an intermediate processing point. An Extract SPECIALRUN process can read from Oracle GoldenGate or TMF audit trails; a Replicat from Oracle GoldenGate trails.

SPECIALRUN for Extract

You can specify EXTTRAILSOURCE or EXTFILESOURCE for Extract to read data from Oracle GoldenGate trails rather than TMF audit trails.

SPECIALRUN has the following dependencies:

- When you specify SPECIALRUN, you must also specify the BEGIN and END parameters.
- SPECIALRUN must precede EXTFILE, RMTFILE, EXTTRAIL or RMTTRAIL entries.

Syntax

```
SPECIALRUN
[, EXTTRAILSOURCE trail_name]
[, EXTFILESOURCE trail_file_name]
```

EXTTRAILSOURCE *trail_name*
The name of the Oracle GoldenGate trail to use as the data source.

EXTFILESOURCE *trail_file_name*
The name of the Oracle GoldenGate trail containing the single file to use as the data source.

Examples

Example 1

```
SPECIALRUN EXTTRAILSOURCE $data1.ggsdat.et
```

Example 2

```
SPECIALRUN EXTFILESOURCE $data3.etdat.datfile1
```

SPECIALRUN for Replicat

For Replicat you must specify EXTTRAIL, RMTTRAIL, EXTFILE or RMTFILE with a SPECIALRUN.

Replicat always uses BEGIN and END to determine when to start and stop processing for a SPECIALRUN. If nothing is entered for BEGIN, the process will start with the first record. If nothing is entered for END, the process start time will be used to determine the limit of the records to process.

Syntax

```
SPECIALRUN
```

Example

The following is an example of a Replicat parameter file that would be run from a TACL prompt to load records from the file GGSINI.ECUSTMER on the target system to the GGSTAR.TCUSTMER table on the target. The GGSINI.ECUSTMER file was previously loaded to the target by an Extract one-time run.The data definition for GGSSOU.ECUSTMER stored in GGSDEF.ECUSTDEF is used to map the structure of the source records to the target.

```
SPECIALRUN
END RUNTIME
```

```
SOURCEDEFS \B.$DATA3.GGSDEF.ECUSTDEF
EXTFILE \B.$DATA3.GGSINI.ECUSTMER
MAP \A.$DATA1.GGSSOU.ECUSTMER,
TARGET \B.$DATA3.GGSTAR.ECUSTMER;
```

SQLFORMATDISCARDFILE

Valid For

Replicat

Description

Use `SQLFORMATDISCARDFILE` to create a file for SQLCI formatted statements of operations that have been discarded because of errors. Once the original problem is corrected, SQL discard files can be used as input to replicate the failed operations.

The SQL formatted records include `SET SESSION ABORT ERROR ON` to cause the SQL action to abend if there is an error. `BEGIN` and `COMMIT WORK` statements are also included.

The requirements for using the SQL discard file are:

- The target for the Replicat must be SQL/MP. (The source database may be any that Oracle GoldenGate supports.)
- The contents of the SQL formatted discard file can be edited, but must be valid input to SQLCI. Because of the possible size of the file, `TEDIT` should be used; `EDIT` can fail on a memory error.

The SQL discard file is an edit type file so changes can be made if necessary. The size of the file is set at extents (70, 70, 128) and this should not be changed.

> **Note:**
>
> The extents should not be changed because making it 1) smaller may result in an error 45 indicating the file is full when more lines could be processed 2) larger may suppress an error 46 indicating there are more lines than an edit file may contain.

See Using the SQL Formatted Discard File for information on editing the file and the rules that must be followed for valid input to SQLCI.

Default

No SQL discard file created.

Syntax

```
SQLFORMATDISCARDFILE file_name
[, APPEND | PURGE | ROLLOVER]
[, LINECOUNT number]
[, OWNER (group_number, user_number)]
[, SECURE "rwep"]
```

file_name
A physical file name or an existing define name of class map. The *file_name* must include the file location.

APPEND | PURGE | ROLLOVER
Specifies the handling of an existing SQL discard file.

- APPEND - Adds a record to the end of the file.
- PURGE - Purges an existing file and creates a new one. This is the default if no option is entered.
- ROLLOVER - Renames the existing discard file by appending a sequence number to the *file_name* according to these rules:
  - If the *file-name* is 7 characters or less, 1 digit is appended.
  - If the file name is 8 characters, the file will not rollover and a warning is given that the name is too long. This is true when the rollover is requested from GGSCI as well as from the DISCARDFILE parameter.

LINECOUNT *number*
Sets the number of lines in the edit file that will trigger rollover. *number* must be at least 5000 and no greater than 98000. The default is 98000 when no value is entered.

OWNER (*group_number*, *user_number*)
Defines ownership of the discard file.

SECURE "*rwep*"
Secures the file using standard Guardian security for read, write, execute and purge operations.

Example

```
SQLFORMATDISCARDFILE $DATA01.G11RPT.SQLDIS, APPEND, LINECOUNT 24000
```

SQLFORMATDISCARDROLLOVER

Valid For

Replicat

Description

Use SQLDISCARDROLLOVER to specify when a SQL discard file is aged and a new one is created. Old files are renamed in the format of *group_name(n)*, where *group_name* is the name of the Replicat group and*(n)* is a number that gets incremented by one each time a new file is created, for
example: $DATA.ACRPT.SQLDIS0, $DATA.ACRPT.SQLDIS1, $DATA.ACRPT.SQLDIS2, and so forth.

Either the AT or ON option can be entered, or both options can be used together. The following rules apply to the possible variations in entries:

- Entering AT with a time but without a day creates a new discard file at the specified time every day.
- Entering AT without a time generates an error.

- Entering ON with a day but without a time creates a new discard file at midnight on the specified day.
- Entering ON without a day will generate an error.
- If no option is entered, the discard file will roll over at midnight every night.

To have more than one rollover, enter multiple AT, ON, or AT ON options. Up to 30 will be used.

Default

No rules. File is not rolled over unless it reaches the edit file maximum line count of 98000.

Syntax

```
SQLFORMATDISCARDROLLOVER {
AT hh:mm |
ON day-of-week |
AT hh:mm ON day_of_week
}
```

AT *hh:mm*
The time of day to age the file based on a 24-hour clock.
Valid values:

- *hh* is an hour of the day from 1 through 23.
- *mm* is minutes from 00 through 59.

ON *day_of_week*
The day of the week to age the file. Valid values are SUNDAY through SATURDAY. They are not case-sensitive.

Examples

Below are examples of the use of SQLFORMATDISARDROLLOVER.

Example 1
This example closes the existing SQL discard file and creates a new one at 5:30 a.m. each day.

```
SQLFORMATDISCARDROLLOVER AT 05:30
```

Example 2
This example rolls the SQL discard file over every Friday at midnight.

```
SQLFORMATDISCARDROLLOVER ON friday
```

Example 3
This example rolls the SQL discard file over at 5:30 a.m. every Friday.

```
SQLFORMATDISCARDROLLOVER AT 05:30 ON FRIDAY
```

Example 4
This example will roll over the SQL discard file at 5:30 a.m. every Wednesday *and* at 5:30 am every Friday.

```
SQLFORMATDISCARDROLLOVER AT 05:30 ON WEDNESDAY, AT 05:30 ON FRIDAY
```

STATOPTIONS

Valid for

Extract, Replicat

Description

Use `STATOPTIONS` to report zero counts for insert, update, and delete operations. For Replicat also use `STATOPTIONS` to report on the time spent creating a file, purging a file, or ending a transaction.

Default

`ZEROSUPPRESS`, no reporting on Replicat timings

Syntax

```
STATOPTIONS [ZEROSUPPRESS | NOZEROSUPPRESS}[, PROGSTATS]
```

`ZEROSUPPRESS | NOZEROSUPPRESS`
Use to suppress or display zero insert, update, and delete counts.

`PROGSTATS`
Valid for Replicat only. Reports on the time spent creating a file, purging a file, or ending a transaction.

Examples

Example 1
Sample results for `STATOPTIONS NOZERSUPPRESS`:

```
Total # records written to RFUNLDS 349
   \NY015.$DATA06.QABASE.BRANCH    # inserts: 100
                                   # updates: 0
                                   # deletes: 0
   \NY015.$DATA07.LQASRC.HISTORY   # inserts: 249
                                   # updates: 0
                                   # deletes: 0
```

Example 2
Sample results for `STATOPTIONS ZEROSUPPRESS`:

```
Total # records written to RFUNLDS 349
   \NY015.$DATA06.QABASE.BRANCH    # inserts: 100
   \NY015.$DATA07.LQASRC.HISTORY   # inserts: 249
```

Example 3
Sample results for `STATOPTIONS PROGSTATS`:

```
EndTransaction            7  elapsed 00:00:00.079211 PerOp 0
FileCreate                1  elapsed 00:00:00.021796 PerOp 0
FilePurge                 3  elapsed 00:00:00.011630 PerOp 0
```

SUPPRESSALLALTERMESSAGES | NOSUPPRESSALLALTERMESSAGES

Valid for

Extract

Description

Use `SUPPRESSALLALTERMESSAGES` to instruct Extract to suppress all `ALTER` messages.

Default

`NOSUPPRESSALLALTERMESSAGES`

Syntax

```
SUPPRESSALLALTERMESSAGES | NOSUPPRESSALLALTERMESSAGES
```

SUPPRESSALTERMESSAGES | NOSUPPRESSALTERMESSAGES

Valid for

Extract

Description

Use `SUPPRESSALTERMESSAGES` to instruct Extract to display the first `ALTER` message it encounters for a table, but suppress that table's subsequent `ALTER` messages. Secondary partition `ALTER` messages resulting from the initial `ALTER` operation will also be suppressed.

When `SUPPRESSALTERMESSAGES` is set, only one `ALTER` will be seen in the Extract report and EMS for a base table, even if numerous `ALTER` operations are performed on that table during the run of the Extract process.

No `ALTER` messages will be seen for a table that has been excluded for the Extract.

Default

`NOSUPPRESSALTERMESSAGES`

Syntax

```
SUPPRESSALTERMESSAGES | NOSUPPRESSALTERMESSAGES
```

SUPPRESSFETCHCOMPRESSEDDISCARDS | NOSUPPRESSFETCHCOMPRESSEDDISCARDS

Valid for

Extract

Description

Use `SUPPRESSFETCHCOMPRESSEDDISCARDS` to stop the display of messages generated when a `FETCHCOMP` fails to find the needed record. For example, a message like the following will be suppressed:

```
20013-07-16 15:08:58 GGSINFO 213 Compressed record discarded. . .
```

Default

```
NOSUPPRESSFETCHCOMPRESSEDDISCARDS
```

Syntax

```
SUPPRESSFETCHCOMPRESSEDDISCARDS
```

SUPPRESSFILEOPMESSAGES | NOSUPPRESSFILEOPMESSAGES

Valid for

Extract, Replicat

Description

Use `SUPPRESSFILEOPMESSAGES` to suppress the output of messages generated after `FILE RENAME`, `PURGE`, `CREATE`, `ALTER`, `SETMODE`.

Syntax

```
NOSUPPRESSFILEOPMESSAGES
```

SUPPRESSMARKERMESSAGES

Valid for

GLOBALS, Extract, Replicat

Description

Use `SUPPRESSMARKERMESSAGES` to suppress the output of messages generated when markers are processed.

As a global parameter

Enter in the GLOBALS parameter file to suppress messages when markers are processed by Extract, Replicat or using GGSCI commands.

Syntax

```
SUPPRESSMARKERMESSAGES {YES | NO}
```

As a process-specific parameter

Enter in the Extract or Replicat parameter file to suppress messages for that Extract or Replicat process.

Syntax

```
SUPPRESSMARKERMESSAGES
```

SWAPVOL

Valid for

GLOBALS

Description

Use SWAPVOL to designate a swap volume for processes created by Oracle GoldenGate. For SWAPVOL to take effect, you must:

1. Stop Manager with the GGSCI STOP MANAGER command.
2. Exit and restart GGSCI after modifying the GLOBALS parameter file.

Default

The volume on which Oracle GoldenGate is installed

Syntax

SWAPVOL *volume*

volume
Specifies the swap volume.

SYNCFILE

Valid for

Syncfile

Description

Use SYNCFILE when the parameter file has a different name than the group name.

Syntax

Syncfile *group_name*

group_name
The Syncfile group name.

SYSKEYCONVERT

Valid for

Extract

Description

Use `SYSKEYCONVERT` for direct file extraction only. `SYSKEYCONVERT` specifies the format of the syskey in the Oracle GoldenGate output for entry-sequenced and ACI files. Use this parameter only when you are trickling data throughout the day to the target system with `RMTBATCH` and the records are variable length (for example, the ACI BASE24 Super files).

> **Note:**
>
> If `SYSKEYCONVERT` is specified, you cannot mix `RMTBATCH` activity with standard extraction of data for replication.

Default

The Guardian file position

Syntax

```
SYSKEYCONVERT {USEBYTEPOSITION | USERECORDNUMBER}
```

`USEBYTEPOSITION`
Extract uses a byte count from the start of the file as if the input file was an unstructured file. The first byte position is zero. Use this option for the ACI Super files.

`USERECORDNUMBER`
Extract uses a record number from the start of the file of the record. The first record is zero.

TABLE

Valid for

Extract

Description

`TABLE` is a synonym for the `FILE` parameter. The object of the `TABLE` parameter can be a SQL table or view or an Enscribe file.

For a description of the options available with `TABLE`, see "FILE | TABLE".

TABLEEXCLUDE

Valid for

Extract

Description

Use TABLEEXCLUDE with the TABLE or MAP parameter to explicitly exclude tables from being included in a wildcard specification. Must follow a TABLE or MAP statement that uses wildcards.

Syntax

```
TABLEEXCLUDE exclude_specification
```

exclude_specification
The name or wildcard specification of the table to exclude.

Example

In the following example, TABLE retrieves all tables from $DATA.MASTER, except those tables containing the letter X, and the table named BIGTAB. Those tables not excluded by the TABLEEXCLUDE parameter are output to the aa trail. All tables containing the letter X are output to bb trail.

```
RMTTRAIL /ggdat/aa
TABLE $DATA.MASTER.*;
TABLEEXCLUDE $DATA.MASTER.*X*;
TABLEEXCLUDE $DATA.MASTER.BIGTAB;

RMTTRAIL /ggdat/bb
TABLE $DATA.MASTER.*X*;
```

TALUSEREXIT

Valid for

Extract, Replicat

Description

Use TALUSEREXIT to call custom TAL routines at different points during processing.

If TALUSEREXIT is specified in the parameter file, but a user exit is not bound to the Extract or Replicat object, the process will abend.

Syntax

```
TALUSEREXIT
```

TCPBUFSIZE

Valid for

Extract

Description

Use `TCPBUFSIZE` to set the maximum size of the message buffer sent to the Collector process over TCP/IP. The valid range for `TCPBUFSIZE` is 0 through 65519.

> **Note:**
>
> Extract will lower the default and maximum to 28000 bytes if the `RMTTASK` parameter is used or if the Collector is not release 11.1.1 or later.

`TCPFLUSHBYTES` affects `TCPBUFSIZE` so review your setting for `TCPFLUSHBYTES` when you set `TCPBUFSIZE`. For example, if `TCPFLUSHBYTES` is set to 8000, even though you set `TCPBUFSIZE` to 65000, the buffer will be flushed each time it reaches 8000 bytes.

Default

65519

Syntax

`TCPBUFSIZE` *`bytes`*

`bytes`
The buffer size in bytes.

TCPFLUSHBYTES

Valid for

Extract

Description

Use `TCPFLUSHBYTES` to set the size of the record buffer that Extract sends to remote systems. Depending on the remote system, changing this parameter can have a significant impact on performance. Valid values are `0` to `65519`.

> **Note:**
>
> Extract will lower the default and maximum to `28000` bytes if the `RMTTASK` parameter is used or if the Collector is not release 11.1.1 or later.

Review your setting for TCPBUFSIZE when you set TCPFLUSHBYTES. Because TCPBUFSIZE is the maximum buffer size, it limits TCPFLUSHBYTES. For example, if TCPBUFSIZE is set to 8000 bytes, TCPFLUSHBYTES cannot exceed 8000 bytes regardless of its setting.

Default

65519

Syntax

TCPFLUSHBYTES *bytes*

bytes
The buffer size in bytes.

TCPIPPROCESSNAME

Valid for

Manager, Extract

Description

Use TCPIPPROCESSNAME to specify a TCP/IP process other than the default. For Manager this is the process it uses to listen for requests coming from other systems. For Extract this specifies the process for data transfer.

Selection Hierarchy for Manager

Manager selects the option to use based on this hierarchy:

1. Stand-alone parameter TCPIPPROCESSNAME entry in the Manager parameter file
2. DEFINE =TCPIP^PROCESS^NAME added before the Manager process was started
3. GLOBALS entry for TCPIPPROCESSNAME
4. Default $ZTC0

Selection Hierarchy for Extract

There are other ways to specify the TCPIPPROCESSNAME when the NonStop TCP/IP process name is not $ZTC0. When assigning a process name, Extract selects the option to use based on this hierarchy:

1. RMTHOST TCPIPPROCESSNAME option in the Extract parameter file
2. Stand-alone parameter TCPIPPROCESSNAME entry in the Extract parameter file before the RMTHOST entry.
3. DEFINE =TCPIP^PROCESS^NAME added before the Extract process was started
4. GLOBALS entry for TCPIPPROCESSNAME
5. MGR's entry for TCPIPPROCESSNAME when process is started from GGSCI
6. Default $ZTC0

You can include additional TCPIPPROCESSNAME entries for Extract. With a RMTHOSTALT address and TCPIPSWITCHERRS, Extract will change both the backup TCP/IP process and the destination IP address after the specified number of retries. Refer to RMTHOSTALT on page "RMTHOSTALT" for an example.

Default

`$ZTC0`

Syntax

`TCPIPPROCESSNAME process_name [@ip_address]`

process_name
A valid HP NonStop TCP/IP process name.

ip_address
Restricts the process to the specified IP address.

Examples

Example 1

The following Extract parameter setting will cause data to be output over TCP/IP process `$ZTC4` instead of the default `$ZTC0`.

```
TCPIPPROCESSNAME $ZTC4
```

Example 2

The following Extract parameter setting will cause data to be output over TCP/IP process `$ZTC4` using the IP address `2001:db8:2010:5040:4fff:ffff:ffff:28`.

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

TCPSOURCETIMER | NOTCPSOURCETIMER

Valid for

Extract

Description

Use `TCPSOURCETIMER` or `NOTCPSOURCETIMER` to specify whether the Collector adjusts the timestamps of each record from the source system to the timestamps of the target system. When transmitting data to trails or files (`RMTFILE` or `RMTTRAIL`) on the target system using the TCP/IP Collector, the default is to adjust the timestamps. Although timestamps are stored in Greenwich Mean Time (GMT), the clocks of the source and target systems are frequently un-synchronized. Adjusting those timestamps allows Manager and GGSCI to calculate the true latency between the source and target systems.

However, when data is moved over TCP/IP between two NonStop systems and the systems are also linked with an Expand network, this adjustment is already made. Under those conditions, specify `NOTCPSOURCETIMER` to avoid the incorrect "double adjustment." `NOTCPSOURCETIMER` should be used when a Replicat on the target system uses `@GETENV` to get the `COMMITTIMESTAMP` header field for any column mapping.

Default

`TCPSOURCETIMER`

Syntax

```
TCPSOURCETIMER | NOTCPSOURCETIMER
```

THRESHOLD | NOTHRESHOLD

Valid for

Manager

Description

Use `THRESHOLD` to generate an event message when the percentage of all audit left on disk or tape falls below the specified percentage. Use `NOTHRESHOLD` to eliminate all threshold reporting.

See "DISKTHRESHOLD | NODISKTHRESHOLD" for a description of the `DISKTHRESHOLD` parameter and audit threshold processing.

Default

```
THRESHOLD 20
```

Syntax

```
THRESHOLD percent_left | NOTHRESHOLD
```

percent_left
The specified percentage.

TMFDUMPAGE

Valid for

`GLOBALS`, Manager

Description

`TMFDUMPAGE` limits information returned by the `TMFDUMPINFO` command to dumps that are created during the specified number of days.

Default

30

Syntax

```
TMFDUMPAGE number_days
```

number_days
The number of days for limiting TMF dump information.

TMFDUMPTABLEENTRIES

Valid for

GLOBALS, Manager

Description

TMFDUMPTABLEENTRIES limits information returned by TMFDUMPINFO to the number of specified dumps.

Default

1024

Syntax

TMFDUMPTABLEENTRIES *max_dumps*

max_dumps
The number of dump entries to display. The maximum allowed is 6000.

TMFEXCEPTIONS | NOTMFEXCEPTIONS

Valid for

Replicat

Description

Use TMFEXCEPTIONS when the application performs multiple operations on the same record within the same transaction. Certain application scenarios can cause out-of-order records in the TMF audit trail within a given transaction:

- An insert on a record can occur in the audit trail before a delete on the same primary key, even though the application performed the delete first, followed by the insert (resulting in a duplicate record condition when the insert is performed)
- An update can occur in the audit trail before the insert on the same primary key (resulting in a missing record error when the update is performed)

TMFEXCEPTIONS resolves these conditions.

In the event of a duplicate insert, TMFEXCEPTIONS saves the duplicated insert until the end of the transaction. If a delete with the same primary key is subsequently encountered, the delete is performed, then the saved insert is performed.

In the event of a missing update record, TMFEXCEPTIONS saves the missing update until the end of the transaction. If an insert with the same primary key is subsequently encountered, the insert is performed, then the saved update is performed.

Use NOTMFEXCEPTIONS to turn off the exception processing.

TMFEXCEPTIONS cannot be used with INSERTMISSINGUPDATES.

Default

NOTMFEXCEPTIONS

Syntax

TMFEXCEPTIONS | NOTMFEXCEPTIONS

Example

This example turns on TMFEXCEPTIONS for ORDERS but toggles it off for any MAP statements that are defined later.

```
TMFEXCEPTIONS
MAP $DATA1.SQLDAT.ORDERS, TARGET $DATA1.MASTER.ORDERS;
NOTMFEXCEPTIONS
```

TMFREFRESHINTERVAL

Valid for

GLOBALS, Manager, Extract

Description

Use TMFREFRESHINTERVAL to set the refresh interval in seconds.

The refresh interval can be set in the either the GLOBALS or Manager parameter, but should not be set in both. The GLOBALS or Manager file setting can be overridden by:

- TMFREFRESHINTERVAL in Extract.
- Executing the Oracle GoldenGate TMF commands in GGSCI (see "" on page 1-44).

Default

15

Syntax

TMFREFRESHINTERVAL *minutes*

minutes
The refresh interval in minutes.

TMFTRAILTRACE

Valid for

Extract

Description

Use to tell the TMFTRAILTRACE program to write messages to the report file when it is checking for the next trail.

Syntax

```
TMFTRAILTRACE
```

TRACE

Valid for

Syncfile

Description

Use `TRACE` for debugging purposes. `TRACE` logs messages regarding Syncfile processing to the report file. You can use the report file to help debug Syncfile issues, generally during an Oracle GoldenGate support session.

Syntax

```
TRACE
```

TRACECLOSES

Valid for

Logger

Description

Use `TRACECLOSES` to send close records to Logger for the purpose of tracing activity. This parameter captures all open attempts on the list of files that `GGSLIB` has been configured to capture.

Use the Logdump utility to examine these records and correlate them to Logger activity.

Default

Do not trace.

Syntax

```
TRACECLOSES
```

TRACEOPENS

Valid for

Logger

Description

Use `TRACEOPENS` to send open records to Logger for tracing system activity. This parameter captures all open attempts on the list of files that `GGSLIB` has been configured to capture.

Use the Logdump utility to examine these records and correlate them to Logger activity.

Default

Do not trace.

Syntax

```
TRACEOPENS
```

TRACEPROCESSIOS

Valid for

Logger

Description

Use `TRACEPROCESSIOS` to instruct Logger to precede each logged record with information regarding the process that created the record. Use this information to determine and trace the process, program, and user that handle particular updates to the application database. If `TRACEPROCESSIOS` is on, a separate trace record is added to the log trail, which creates a small amount of system overhead.

Use the Logdump utility to examine these records and correlate them with database operations. This parameter applies only to the current log. When you specify `TRACEPROCESSIOS`, `TRACESTATS` is automatically turned on.

Default

Do not trace.

Syntax

```
TRACEPROCESSIOS
```

TRACESTATS

Valid for

Logger

Description

Use `TRACESTATS` to instruct Logger to record statistics for each process that sends it operations. When the process stops, or upon a `SEND LOGGER FLUSHSTATS` command, process statistics are output to the log trail.

Statistics include the number of file inserts, updates, deletes, and other information. Use the Logdump utility to examine these statistics. This parameter applies only to the current log. If `TRACEPROCESSIOS` is specified, `TRACESTATS` is automatically turned on.

Default

Do not trace.

Syntax

```
TRACESTATS
```

UPDATEDELETES | NOUPDATEDELETES

Valid for

Replicat

Description

Use `UPDATEDELETES` to convert delete records to update operations. `UPDATEDELETES` applies to all maps specified below it in the parameter file until `NOUPDATEDELETES` is specified.

Default

```
NOUPDATEDELETES
```

Syntax

```
UPDATEDELETES | NOUPDATEDELETES
```

UPDATEINSERTS | NOUPDATEINSERTS

Valid for

Replicat

Description

Use `UPDATEINSERTS` to change insert operations in the target to update operations.

Default

```
NOUPDATEINSERTS
```

Syntax

```
UPDATEINSERTS | NOUPDATEINSERTS
```

UPREPORT

Valid for

Manager

Description

Use `UPREPORTMINUTES` or `UPREPORTHOURS` to periodically report Extract and Replicat process that are running. By default, running processes are not reported, but events are automatically generated when a process is started or stopped.

Default

Do not report running processes

Syntax

```
UPREPORTMINUTES minutes | UPREPORTHOURS hours
```

minutes
The reporting interval in minutes.

hours
The reporting interval in hours.

VERBOSE

Valid for

Syncfile

Description

Use VERBOSE to display FUP and TACL messages. By default, Syncfile does not display the output from FUP or TACL during duplication, but produces summary messages describing the number of files duplicated at any given time.

VERBOSE follows the DUP parameter statement that specifies the FUP or TACL options.

Default

Do not display output.

Syntax

```
VERBOSE;
```

VERIDATAREPORTAGE

Valid for

Manager

Description

Use the VERIDATAREPORTAGE parameter to purge old Veridata report files when they have reached the age limit.

Default

Seven days (7 DAYS);

Syntax

```
VERIDATAREPORTAGE duration unit_of_time
```

The `duration` is specified as a positive integer, followed by the unit of time in any of the following formats to indicate seconds, minutes, or hours. Do not put a space between the numeric value and the unit of time. The unit is not case-sensitive.

```
SEC|SECOND|SECONDS
MIN|MINUTE|MINUTES
HOUR|HOURS|HRS
DAY|DAYS
```

Example

```
VERIDATAREPORTAGE 5 HOURS
```

WAITFILEEVENT

Valid for

Extract, Replicat

Description

Use `WAITFILEEVENT` to wait for a specific event before proceeding. Bi-directional scenarios in particular may need to wait for an external event before proceeding.

For example, a file called `SOURCE.FLTODAY` is mapped to a file called `TARGET.FLTODAY`. Every day at 17:00, `SOURCE.FLTODAY` is renamed to `SOURCE.FLyymmdd`. To propagate the data from `SOURCE.FLTODAY` to the appropriate target file, a `WAITFILEEVENT` command is used.

With `WAITFILEEVENT`, Replicat delays upon receiving information that `SOURCE.FLTODAY` has been created, until `TARGET.FLTODAY` is also created.

`WAITFILEEVENT` produces event messages (which also appear in the Replicat report file) regarding Replicat status while waiting. To bypass the wait event, issue the GGSCI command: `SEND REPLICAT` *`group_name`*, `BYPASSFILEEVENT`

To use `WAITFILEEVENT`, the source file must be specified in a corresponding `MAP`.

Syntax

```
WAITFILEEVENT source_filespec
[, CREATES]
[, CREATIONWINDOW unit num_units]
[, EXEC "command"]
[, PURGES]
[, RENAMES]
[, WARNAFTER unit warn_time]
```

`source_filespec`
The file name or wildcard name to which the wait rules apply.

`CREATES`
Wait until *`source_filespec`* exists. If *`source_filespec`* does not exist in the `CREATIONWINDOW` time period, Replicat keeps checking every 10 seconds until the file has been created.
You must specify one or more `CREATES` to indicate the events on which to wait.

CREATIONWINDOW *unit num_units*
The amount of time Replicat checks for the new file.

- *unit* can be SECONDS, MINUTES, HOURS; for example CREATIONWINDOW HOURS.
- *num_units* can be any positive integer.

The default is CREATIONWINDOW HOURS 20.

EXEC "*command*"
Execute any valid TACL command or program instead of processing the specified CREATES, PURGES, RENAMES, or some other event. The success of the command is not evaluated. If error handling is necessary, implement it within the command or program being executed.
When you specify EXEC, CREATIONWINDOW and WARNAFTER have no effect.

PURGES
Wait until *source_filespec* does not exist.
You must specify one or more of PURGES to indicate the events on which to wait.

RENAMES
Wait until *source_filespec* is renamed. If *source_filespec* has not been renamed in the CREATIONWINDOW time period, Replicat keeps checking every 10 seconds until the file has been renamed.
You must specify one or more RENAMES to indicate the events on which to wait.

WARNAFTER *unit warn_time*
The amount of time before a critical event is issued to EMS indicating that the event has not been satisfied.

- *unit* can be SECONDS, MINUTES, HOURS; for example WARNAFTER HOURS.
- *warn_time* can be any positive integer.

Examples

Example 1
The following example demonstrates the use of CREATIONWINDOW, WARNAFTER, CREATES, and RENAMES.

```
WAITFILEEVENT \GGS.$DATA1.DAT.ACCOUNT*, &
CREATIONWINDOW HOURS 5, &
WARNAFTER MINUTES 30, &
CREATES, RENAMES
```

Example 2
The following example uses the EXEC option to rename a target file after the source has been purged.

```
WAITFILEEVENT $data1.dat.account, PURGE, &
EXEC "rename $data2.target.account, &
$data2.targold.account"
```

Y2KCENTURYADJUSTMENT | NOY2KCENTURYADJUSTMENT

Valid for

Extract, Replicat

Description

Use `Y2KCENTURYADJUSTMENT` to enable the Y2K window to change the century. By default the column conversion functions that deal with dates put a Y2K window on a two-digit year when the century is zero. If the two-digit year is greater than `49`, then the century becomes `19`. If the year is `49` or less, the century becomes `20`. This damages input date times such as `0001-01-01:00:00:00.000000` by making the century `20`, and producing the incorrect result: `2001-01-01:00:00:00.000000`.

Use `NOY2KCENTURYADJUSTMENT` to prevent the Y2K window from changing the century. Enter before relevant date calculations.

`NOY2KCENTURYADJUSTMENT` does not have an effect if the century field is not present at all. When the century field is not present, the century will be assigned a value based on the year. When the year is greater than `49`, the century is set to `19`; otherwise it will default to `20`.

Default

```
Y2KCENTURYADJUSTMENT
```

Syntax

```
Y2KCENTURYADJUSTMENT | NOY2KCENTURYADJUSTMENT
```

3
Collector Parameters

Learn about collector parameters, which you use to control and modify collector operations.

> **Note:**
>
> Collector parameters are specified by a dash (-) followed by a letter and optional arguments. The Collector Parameters are case-sensitive.

Collector Parameters

Most Collector parameters are entered in the Extract parameter file, using the `PARAMS` *`collector_params`* option of the `RMTHOST` parameter. The Manager automatically sends some of these parameters when it starts the Collector, such as -k and `ENCRYPT` key and `KEYNAME` options entered by the user.

The Collector parameters discussed here are for a Collector running on a remote NonStop system. For Collector parameters that run on other platforms, see Collector Parameters.

`-PROCESS` ***`process_name`***
The name of the NonStop TCP/IP process.

`-BULKIO`
Results in Collector writing data in 28K blocks, rather than 4K. Setting `-BULKIO` can increase total throughput of the Collector and reduce overhead significantly.

`-BULKIOLEN`
Use `BULKIOLEN` to customize the block size Collector should use when writing data. Increasing the block size used by Collector can contribute a net performance gain by increasing total throughput. Block sizes may be indicated in bytes or kilobytes, but the Collector only recognize sizes that are multiples of 2. If you choose a block size between 2 and the next multiple, `BULKIOLEN` automatically rounds your specification down.

`-b` ***`backup_cpu`***
Runs the Collector backup process started in *`backup_cpu`*. This option is only for static servers. It is not valid for dynamic servers, which are persistent processes that will be restarted if the process has a failure.

`-B`
Directs the Collector to buffer files.

`-DEBUG`
Starts the Collector in `DEBUG` mode to allow the user to analyze the actions of the program code.

-EMSEMPHASIS *type*
Controls whether or not emphasis will be turned on for certain types of messages coming from EMSCLNT on open systems.
type can be one of the following:

CRITICAL
Turns emphasis on for CRITICAL or WARNING messages as well as FATAL messages.

FATAL or ERROR
Turns emphasis on only for FATAL messages.

The default is CRITICAL.

-ENCRYPT *encrypt_type*
The type of encryption being passed from the Extract process, as specified with the RMTHOST parameter in the Extract parameter file.
Valid values:

- GGS
- BLOWFISH

If using BLOWFISH, also specify the -KEYNAME option. For more information about Oracle GoldenGate security, see Using Encryption.

-f
Always forces file writes to be flushed to disk before returning a success status to the Extract process.
By default, the file system buffers the I/O because it is more efficient than flushing to disk with every operation. Generally, the performance benefits outweigh the small risk that data could be lost if the system fails after an I/O is confirmed successful, but before the buffer actually is flushed to disk. Use -f if this risk is unacceptable, with the understanding that it can compromise the performance of Oracle GoldenGate.

-HOMETERM *terminal*
Identifies the home terminal for the Collector.

-k
Directs Collector to terminate when the Extract process that it is serving disconnects. This option is used by the Manager process when starting the Collector process.

-KEYNAME *name*
Specifies a key name defined in the local ENCKEYS lookup file. *name* is the name of a key in a lookup file on the target system. For more information about Oracle GoldenGate security, see Using Encryption.

-l *file_name*
Directs the Collector to log to *file_name*. The name should be fully qualified.

-m *number*
Specifies the maximum number of files to allocate.

-NOFLUSHCACHE | FLUSHCACHE
Turns on or off the flushing of the disk cache.

-p *port_number*
A TCP/IP port number on which the Collector listens for connections from the Extract process. The default is port `7819`.

-t
Turns on TCP tracing to write detail to EMS on socket operations. It is recommended that you use this parameter only when requested by Oracle GoldenGate Support.

-TCPSTATS
Specifies that TCP statistics should be printed at file rollover.

-v
Specifies that NonStop Collector processes use EMS logging.

QSAM Configuration Parameters

For OS/390 QSAM files, both fixed-length and variable-length, you can specify a configuration file on the OS/390 platform to set certain file creation characteristics. Indicate the location of this file with the `-P` collector parameter.

You can also specify QSAM file creation characteristics using the `RMTBATCH` Extract program parameter. The syntax is:

```
RMTBATCH file_spec, option [, option, ...];
```

file_spec
A file name or wildcard for the incoming file. Any files that match *`file_spec`* are assigned the file creation rules you specify with *`option`*. If a file name is satisfied by more than one `RMTBATCH` entry in the configuration file, the last entry is used.

option
One of the following.

CLOSEACTION "*system_command*"
Submits a shell script or system command to the system upon file close, when close was a result of completed processing of the corresponding source file (indicated by a "file close on source" message).
The Collector waits for the command to complete. To return control to the Collector immediately, specify a "nowait" type operation in the script or command (as with the `&` in UNIX).

COMMENT
Lets you insert comments. Anything on the same line after `COMMENT` is ignored. Comments can also begin with two hyphens (--).

SPACE (*primary_quantity*, *secondary_quantity*, *unit*)
Mandatory parameter. Requests space for a new data set.

primary_quantity
The number of tracks, cylinders, or blocks to be allocated for a new data set.

secondary_quantity
The number of additional units to allocate if more space is needed.

unit
The units in which space is requested. Use `TRK` for tracks, `CYL` for cylinders, or `BLK` for blocks.

`TARGETFTYPE {QSAMV | QSAMF}`
Mandatory parameter for QSAM files. Specify `QSAMV` for variable-length MVS QSAM files or `QSAMF` for fixed-length MVS QSAM files.

`TARGETRECLEN` ***`record_length`***
Mandatory parameter for QSAM files. Specify the length in bytes of a fixed-length record or the maximum length in bytes for variable-length records. *`record_length`* can be from `1` to `32760`.

`TARGETBLKSIZE` ***`block_length`***
Parameter for QSAM files. Specify a multiple of the record length for fixed-length records, or the maximum record length +4 for variable-length records. *`block_length`* can be from 1 to 32760.

`VOLUME` ***`serial_number`***
Mandatory parameter. Identifies the serial number of the volume on which the target data set will reside. Specify only one volume serial number.

Example

The following example matches all source file names. It creates fixed length records of `100` bytes, in blocks of `1000` bytes (10 records per block), with a volume serial number for the new data set of `123456`, and allocates an initial space of `20` tracks. If more space is needed, it is allocated in increments of `10` tracks.

```
RMTBATCH *,
TARGETFTYPE QSAMF,
TARGETRECLEN 100, TARGETBLKSIZE 1000,
VOLUME 123456, SPACE (20,10,TRK),
CLOSEACTION "type * %";
```

4
Field Conversion Functions

Learn how to use Oracle GoldenGate field conversion functions. These functions allow you to manipulate numbers, strings and source columns or field values into the appropriate format for target columns.

- Working with columns
- Working with dates
- Working with numbers and arithmetic expressions
- Working with conditional statements
- Working with character and numerical strings
- Working with stored procedures
- Returning error and lag information

Overview of Functions

This section provides an overview of Oracle GoldenGate functions for the NonStop platform.

Function Syntax

Field conversion functions use the following general syntax:

```
@FUNCTION (expression)
```

`@FUNCTION`
The function name, such as `@DATE` or `@IF`.

`(expression)`
The operations for the function to perform. The operations depend on the function.

Function Summaries

This section summarizes the Oracle GoldenGate functions for NonStop, based on their functionality. An alphabetized reference of the functions follows this section.

Working with Columns

Function	Description
COLSTAT	Returns whether a column is missing, `NULL` or an invalid value.
COLTEST	Tests whether a column is present in a record, missing, `NULL` or an invalid value.

Function	Description
VALONEOF	Returns TRUE if a column contains one of a list of values.

Working with Dates

Function	Description
DATE	Returns a date from a variety of sources in a variety of output formats.
DATEDIFF	Returns the difference between two dates or datetimes, in days or seconds
DATENOW	Returns the current date and time.

Working with Numbers and Arithmetic Expressions

Function	Description
COMPUTE	Returns the result of an arithmetic expression.
CONVERTFLOAT	Converts Tandem float numbers from the HP NonStop to and from the IEEE format used for Windows and UNIX.

Working with Conditional Statements

Function	Description
CASE	Selects a value depending on a series of value tests.
EVAL	Selects a value depending on a series of independent tests.
GETVAL	Extracts parameters from a stored procedure as input to a FILTER or COLMAP clause.
IF	Selects one of two values depending on whether a conditional statement returns TRUE or FALSE.

Working with Character and Numeric Strings

Function	Description
BINARY	Keeps source data in its original binary format in the target column when the source column is defined as a character column.
HIGHVAL | LOWVAL	Emulates the COBOL high and low value functions. Sets COBOL-type group level to either a high or low values when specified conditions are met.

Function	Description
NUMBINNUMBIN	Converts a binary string into a number (for example, a 48-bit Himalaya timestamp).
NUMSTR	Converts a string into a number.
STRCAT	Concatenates one or more strings.
STRCMP	Compares two strings and returns a result of less than, equal, or greater than.
STREQ	Compares two strings and returns a Boolean result of equal or not equal.
STREXT	Extracts selected characters from a string.
STRFIND	Finds the occurrence of a string within a string.
STRLEN	Returns the length of a string.
STRLTRIM	Trims leading spaces in a column.
STRNCAT	Concatenates one or more strings up to a limited number of characters per string.
STRNCMP	Compares two strings up to a certain number of characters.
STRNUM	Converts a number into a string, with justification and zero-fill options.
STRRTRIM	Trims trailing spaces in a column.
STRSUB	Substitutes one string for another within a column.
STRTRIM	Trims leading and trailing spaces in a column.
STRUP	Changes a string to uppercase.

Environmental Information

Function	Description
GETENV	Returns information about the Oracle GoldenGate environment.

BINARY

Use the `@BINARY` function when a source column referenced by a column-conversion function is defined as a character column but contains binary data that must remain binary on the target. By default, once a column is referenced by a column function, the data is converted (if necessary) to ASCII and assumed to be a null terminated string. The `@BINARY()` function copies arbitrary binary data to the target column.

Syntax

```
@BINARY (column_name)
```

Example

This example shows how the binary data in the source column `ACCT_CREATE_DATE` will be copied to the target column `ACCT_CHIEF_COMPLAINT`.

```
MAP \PROD.$DATA1.FINANCE.ACCTOLD, TARGET $DATA01.REPT.ACCT,
COLMAP (USEDEFAULTS,
ACCT-CHIEF-COMPLAINT = @IF (@NUMBIN (ACCT-CREATE-DATE) < 12345, "xxxxxx",
@BINARY(ACCT-CHIEF-COMPLAINT)
);
```

CASE

Allows the user to select a value depending on a series of value tests. There is no practical limit to the number of cases; however, for numerous cases, it is beneficial to list the most frequently encountered conditions first.

Syntax

```
@CASE (value, test_value1, test_result1
[, test_value2, test_result2] [, ...] [, default_result])
```

value
The column you are testing values for.

test_value1
The value to test against the value you are reading.

test_result1
The result to return.

default_result
The result returned when *test_values* are not entered.

Examples

Example 1

The following returns "A car" if `PRODUCT_CODE` is `"CAR"` and "A truck" if `PRODUCT_CODE` is `"TRUCK"`. In this case, if `PRODUCT_CODE` fits neither of the first two cases, a `FIELD_MISSING` indication is returned.

```
@CASE (PRODUCT_CODE, "CAR", "A car", "TRUCK", "A truck")
```

Example 2
In this modified case, assuming PRODUCT_CODE is neither "CAR" nor "TRUCK", "A vehicle" is returned.

```
@CASE (PRODUCT_CODE, "CAR", "A car", "TRUCK", "A truck", "A vehicle")
```

COLTEST

Use COLTEST to perform conditional calculations. COLTEST can check for one or more of the following column conditions and returns TRUE if one of the following column conditions are met.

- PRESENT, which indicates a column is present in the source record and not null. In a compressed record, columns may be missing, but this not the same as null.
- NULL, indicating the column is present and NULL.
- MISSING, indicating that the column is not present.
- INVALID, indicating the column is present but contains invalid data. For example, a PIC 9(3) field that contains spaces yields an INVALID condition.

Syntax

```
@COLTEST (source_field, test_item [, test_item] [, ...])
```

source_field
The name of the field or column that is the source of the data being tested.

test_item
One of: PRESENT, MISSING, INVALID, or NULL.

Examples

Example 1
This example shows how you can calculate the value of a HIGH_SALARY column only if the SALARY field in the source record is both present and greater than a certain number. Set up a test condition with the @IF function to return the result of SALARY when part of the current record and exceeding 250000, otherwise return NULL:

```
HIGH_SALARY = @IF(@COLTEST(BASE_SALARY, PRESENT) AND BASE_SALARY > 250000,
    BASE_SALARY, @COLSTAT(NULL))
```

In this example, the condition BASE_SALARY > 250000 is evaluated only when SALARY is present in the source record and not null. If the presence of the column was not tested first, the column would not have been mapped, because the result would have been missing.

Example 2
In the following example, 0 is returned when AMT field is missing or invalid, otherwise AMT is returned.

```
AMOUNT = @IF (@COLTEST (AMT, MISSING, INVALID), 0, AMT)
```

COMPUTE

@COMPUTE returns the value of an arithmetic expression to a target column. The value returned from the function is in the form of a string.

You can omit `@COMPUTE` when returning the value of an arithmetic expression to another function.

Syntax

```
@COMPUTE (value operator value)
```

value
One or more values on which you are performing calculations.

operator
A valid arithmetic or logical operator.

Examples

Example 1
This example adds `AMT` and `AMT2` and returns the total to `AMOUNT_TOTAL`.

```
AMOUNT_TOTAL = @COMPUTE (AMT + AMT2)
```

Example 2
This example is invalid because the expression is not enclosed in parentheses.

```
AMOUNT_TOTAL = AMT + AMT2
```

CONVERTFLOAT

`@CONVERTFLOAT` converts HP NonStop Tandem (TDM) float numbers to Windows and UNIX Institute of Electrical and Electronics Engineers (IEEE) format or from IEEE float format to TDM float format.

The `@CONVERTFLOAT` function always converts if you map the column; it does not check whether it is a 32 or 64 bit float data type. However, it is not necessary or wise to use `@CONVERTFLOAT` to map 32 bit float columns for conversion to IEEE, because Oracle GoldenGate for Windows and UNIX automatically does this conversion.

> **Note:**
>
> To use `@CONVERTFLOAT`, the Extract and Replicat must be native objects; not TNS.

Syntax

```
@CONVERTFLOAT (col_name, (TOIEEE | TOTDM))
```

col_name
The name of the column containing the float numbers to be converted.

`TOIEEE | TOTDM`
The format to which to convert; IEEE or TDM.

Example

```
COLMAP num01 = @CONVERTFLOAT (num01, TOIEEE)
```

DATE

`@DATE` returns dates and times in a variety of formats to the target column based on the format passed into the source. `@DATE` converts virtually any type of input into a valid SQL date. `@DATE` can also be used to extract portions of a date field, or to compute a numeric timestamp field based on a date.

Syntax

```
@DATE ("output_descriptor", "input_descriptor", source_field
[, "input_descriptor", source_field] [, ...])
```

output_descriptor
A string containing date descriptors and optional literal values. For example, the descriptor `YYYY` corresponds to a four-digit year, the descriptor `MI` describes minutes, while spaces, colons or other literals are output as is. See "Date Descriptors" for descriptions.

input_descriptor
A string containing a series of date descriptors and optional literal values. For example, the descriptor `YYYY` corresponds to a four-digit year, the descriptor `MI` describes minutes. Date descriptors are strung together to describe the field or column that follows in the next parameter. See "Date Descriptors" for descriptions.

source_field
The name of a source field supplying the preceding input.

Date Descriptors

Descriptor	Description
`CC`	Century
`YY`	Two-digit year
`YYYY`	Four-digit year
`MM`	Numeric month
`MMM`	Alphanumeric month, such as `APR`, `OCT`
`DD`	Numeric day of month
`DDD`	Numeric day of the year, such as `001`, or `365`
`DOW0`	Numeric day of the week where Sunday = `0`.
`DOW1`	Numeric day of the week where Sunday = `1`.

Descriptor	Description
DOWA	Alphanumeric day of the week, such as SUN, MON, TUE
HH	Hour
MI	Minute
SS	Seconds

JTSLCT	Use for a Julian timestamp that is already local time, or to keep local time when converting to a Julian timestamp.
	An example of a 48-bit NonStop to 64-bit Julian for LCT to LCT time is:
	`date = @date ("JTSLCT", "TTS", @numbin(date));`
	An example of a NonStop 64-bit Julian for LCT to date type in Oracle:
	`date = @date ("YYY-MM-DD HH:MI:SS", "JTSLCT", date);`

JTSGMT	Julian timestamp, the same as JTS
JTS	Julian timestamp. For more information see "Using JUL and JTS".
JUL	Julian day
TTS	NonStop 48-bit timestamp
PHAMIS	PHAMIS application date format
FFFFFF	Fraction (up to microseconds)
STRATUS	STRATUS application timestamp that returns microseconds since 1/1/1980.
CDATE	C timestamp in seconds since the Epoch.

Using JUL and JTS

JUL and JTS produce numbers you can use in numeric expressions.

Example

The following expression produces the time at which an order is filled.

```
ORDER_FILLED = @DATE ("YYYY-MM-DD:HH:MI:SS", "JTS", @DATE
("JTS", "YYMMDDHHMISS", ORDER-TAKEN-TIME) + ORDER-MINUTES * 60 * 1000000)
```

The above expression changes ORDER-TAKEN-TIME into a Julian timestamp, then adds ORDER-MINUTES converted into microseconds to this timestamp (the inner @DATE

expression). This expression is passed back as a new Julian timestamp to the outer `@DATE` expression, which converts it back to a more readable date and time.

Working with Date Strings

Descriptor string `"YYYYMMDD"` indicates that the following numeric or character field contains (in order) a four-digit year (`YYYY`), month (`MM`), and day (`DD`).

Descriptor string `"DD/MM/YY"` indicates that the field contains the day, a slash, the month, a slash, and the two digit year.

Converting Two-digit Years into Four-digit Values

In an instance where a two-digit year is supplied, but a four-digit year is required in the output, several options exist.

- A century is hard-coded, as in the `"CC", 19` or `"CC", 20`.
- The `@IF` function is used, as in `"CC", @IF (YY > 70, 19, 20)`. This causes century to be set to `19` when year is greater than `70`, otherwise `20`.
- The system calculates the century automatically. If the year is less than `50`, the system calculates a century of `20`; otherwise, the century calculates to `19`.

Example

The following show some ways to use date conversions.

- Converting year, month and day fields into a SQL date.

  ```
  DATE_COL = @DATE ("YYYY-MM-DD", "YY", date1.yr, "MM", date1.mm, "DD", date1.dd)
  ```

- Converting the date at the group level (assuming year, month, and day are part of `date1`).

  ```
  DATE_COL = @DATE ("YYYY-MM-DD", "YYMMDD", date1)
  ```

- Converting to a date and time, defaulting seconds to zero.

  ```
  DATE_COL = @DATE ("YYYY-MM-DD:HH:MI:00", "YYMMDD", date1, "HHMI", time1)
  ```

- Converting a numeric field stored as `YYYYMMDDHHMISS` to a SQL date.

  ```
  DATETIME_COL = @DATE ("YYYY-MM-DD:HH:MI:SS", "YYYYMMDDHHMISS", NUMERIC-DATE)
  ```

- Converting a numeric field stored as `YYYYMMDDHHMISS` to a Julian timestamp.

  ```
  JULIAN_TS_COL = @DATE ("JTS", "YYYYMMDDHHMISS", NUMERIC-DATE)
  ```

- Converting a Julian timestamp field to two columns: a datetime field in the format `YYYYMMDDHHMISS`, and a fraction field, which holds the microseconds portion of the timestamp.

  ```
  DATETIME_COL = @DATE ("YYYY-MM-DD:HH:MI:SS", "JTS", JTS-FIELD),
  FRACTION_COL = @DATE ("FFFFFF", "JTS", JTS-FIELD)
  ```

DATEDIFF

`@DATEDIFF` calculates the difference between two dates or datetimes, in days or seconds.

Syntax

```
@DATEDIFF ("difference", date, date)
```

"difference"
The difference between the specified dates. Valid values can be:

- `DD` to compute the difference in days.
- `SS` to compute the difference in seconds.

date
A string in the format of `YYYY-MM-DD[*HH:MI[:SS]]`, where * can be a colon `(:)` or a blank space.

Examples

Example 1
To calculate the number of days since the beginning of the year `2000`:

```
differential = (@DATEDIFF("DD","2000-01-01",@DATENOW()))
```

Example 2
To calculate the actual day of the year in the above example (`@DATEDIFF` returns `0` for `2000-01-01`):

```
todays_day = @COMPUTE(@DATEDIFF("DD","2000-01-01",@DATENOW()))+1)
```

EVAL

Allows the user to select a value depending on a series of independent tests. There is no practical limit to the number of conditions. If the number of cases is large, it is beneficial to list the most frequently encountered conditions first.

Syntax

```
@EVAL (condition1, result1 [, condition2, result2] [, ...]
[, default_result])
```

condition
A conditional test using standard conditional operators.

result
A value or string to return based on the results of the conditional test. Enclose literals within double quotes.

default_result
A default result to return if none of the conditions is satisfied. A default result is optional.

Examples

Example 1
In the following example, if `AMOUNT` is greater than 10000, `"high amount"` is returned. If `AMOUNT` is greater than `5000` (and less than or equal to `10000`), `"somewhat high"` is returned (unless the prior condition was satisfied). If neither condition is satisfied, a `COLUMN_MISSING` indicator is returned because a default result is not specified.

```
AMOUNT_DESC = @EVAL (AMOUNT > 10000, "high amount", AMOUNT > 5000,
"somewhat high")
```

Example 2
The following is a modification of the preceding example. It returns the same results, except that a default value is specified, and a result of `"lower"` is returned if `AMOUNT` is less than or equal to `5000`.

```
@EVAL (AMOUNT > 10000, "high amount", AMOUNT > 5000, "somewhat high", "lower")
```

GETENV

`@GETENV` returns a variety of information about Oracle GoldenGate processing, including lag information, the last replicated operation, and Oracle GoldenGate environment information.

Syntax

```
@GETENV (info_type)
```

info_type is one of the following.

`"COMMITTIMESTAMP"`
Returns the timestamp when the transaction was committed as an integer representing the Julian GMT.

`"JULIANTIMESTAMP"`
Returns the current Julian GMT timestamp in the form of an integer such as `2119193858767655 84`.

`"LOCALTIMESTAMP"`
Returns the current system time as an integer representing the Julian LCT.

`("LAG", "unit")`
Returns lag information. See "Reporting Lag Information".

`("LASTERR","option")`
Returns information about the last replicated operation, including detailed error information. See "Returning Information from Replicat".

`("GGENVIRONMENT", "option")`
Returns Oracle GoldenGate environment information. See "Returning Oracle GoldenGate Environment Information".

`("GGFILEHEADER", "option")`
Returns the format and properties of an Oracle GoldenGate trail file, which is stored in the file header record sent from open systems.

`("GGHEADER", "option")`
Returns Oracle GoldenGate record header information. See "Returning Record Header Information".

`("RECORD", "option")`
Returns information about the records that are being processed (such as the sequence number of the trail file), or the name of the source application program that

altered the Enscribe file record. See "Returning Record Location and Source Application Information".

"RECSOUTPUT"
Returns the total number of records processed.

("TLFKEY", SYSKEY *unique_key*)
Enables a unique key to be associated with TLF/PTLF records in the ACI BASE24 application. See "Associating BASE24 Keys and Records".

Reporting Lag Information

Use the "LAG" option of @GETENV to return lag information. Lag is the difference between the time a record was processed by the Extract or Replicat program and the timestamp of that record in the data source. Both LAG and *unit* must be enclosed within double quotes.

Syntax

```
@GETENV ("LAG", "unit")
```

"unit" is one of the following.

"SEC"
Returns the lag in seconds. This is the default when a unit is not explicitly provided for LAG.

"MSEC"
Returns the lag in milliseconds.

"MIN"
Returns the lag in minutes.

Returning Information from Replicat

Use the "LASTERR" option of @GETENV to return information about the last operation processed by the Replicat program. Options provide error information. Both LASTERR and *option* must be enclosed within double quotes.

Syntax

```
@GETENV ("LASTERR", "option")
```

"option" is one of the following.

"DBERRNUM"
Returns the database error number associated with the failed operation.

"DBERRMSG"
Returns the database error message associated with the failed operation.

"OPTYPE"
Returns the operation type that was attempted.

"OSERRNUM"
Specifies a NonStop operating system error.

"ERRTYPE"
Returns the type of error. Possible results are:

- DB (for database errors)
- MAP (for errors in mapping before replicating the record).

Returning Oracle GoldenGate Environment Information

Use the GGENVIRONMENT option of @GETENV to return information about the Oracle GoldenGate environment. This option is valid for the Extract and Replicat program processing.

Syntax

```
@GETENV ("GGENVIRONMENT", "option")
```

"option" is one of the following.

GROUPNAME
Returns the Extract or Replicat group name.

HOSTNAME
Returns the name of the host running the Extract or Replicat programs.

OSUSERNAME
Returns the operating system user name that started the process.

Returning Record Header Information

Use the GGHEADER option of @GETENV to return record header information. This option is valid for the Extract and Replicat processes.

Syntax

```
@GETENV ("GGHEADER", "option")
```

"option" is one of the following.

BEFOREAFTERINDICATOR
Returns the before or after indicator indicating whether the record is a before-image or after-image. Possible results are:

- BEFORE (before-image)
- AFTER (after-image)

COMMITTIMESTAMP
Returns the transaction timestamp (the time when the transaction committed) converted to the local time zone as a string in the format of YYYY-MM-DD HH:MI:SS.FFFFFF, for example:

```
2010-01-24 17:08:59.000000
```

LOGPOSITION
Returns the audit log position.

LOGRBA
Returns the relative byte address for the audit log.

TABLENAME
Returns the table name.

OPTYPE
Returns the type of operation. Possible results are:

- INSERT
- UPDATE
- DELETE
- ENSCRIBE COMPUPDATE
- SQL COMPUPDATE
- PK UPDATE
- TRUNCATE

If the operation is not one of the above types, then the function returns the word TYPE with the number assigned to the type. For more information about possible record types, see the file format information in Using the Logdump Utility.

RECORDLENGTH
Returns the record length.

TRANSACTIONINDICATOR
Returns the transaction indicator. Possible results are:

- BEGIN - Returned when the record header TransInD is 0 indicating the first statement in the transaction.
- MIDDLE - Returned when the header TransInD is 1 indicating a statement in the middle of the transaction.
- END - Returned when the TransInD is 2 indicating the last statement in the transactions
- WHOLE - Returned when the TransInD is 3 indicating only one statement in the transaction.

Returning File Header Information

Use the GGFILEHEADER option of @GETENV to return attributes of an Oracle GoldenGate extract file or trail file that are stored in the file header sent from an Oracle GoldenGate system on Windows or UNIX. Every file in such a trail contains this header. The header describes the file itself and the environment in which it is used.

The file header is stored as a record at the beginning of a trail file preceding the data records. The information that is stored in the trail header provides enough information about the records to enable an Oracle GoldenGate process to determine whether the records are in a format that the current version of Oracle GoldenGate supports.

The trail header fields are stored as tokens, where the token format remains the same across all versions of Oracle GoldenGate. If a version of Oracle GoldenGate does not support any given token, that token is ignored. Deprecated tokens are assigned a default value to preserve compatibility with previous versions of Oracle GoldenGate.

This option is valid for the Replicat process. Both GGFILEHEADER and *return_value* must be enclosed within double quotes.

> **Note:**
>
> If a given database, operating system, or Oracle GoldenGate version does not provide information that relates to a given token, a `NULL` value will be returned.

Syntax

```
@GETENV ("GGFILEHEADER", "return_value")
```

The following sections describe the valid values for *"return_value"*:

- TrailInfo: Information about the trail file
- ProducerInfo: Information about the process that created the trail file
- MachineInfo: Information about the local host of the trail file
- DatabaseInfo: Information about the database that produced the data in the trail file
- ContinuityInfo: Recovery information carried over from the previous trail file

TrailInfo: Information about the trail file

"COMPATIBILITY"
The Oracle GoldenGate compatibility level of the trail file. The compatibility level of the current Oracle GoldenGate version must be greater than, or equal to, the compatibility level of the trail file to be able to read the data records in that file. Current valid values are 0 or 1.

- 1 means that the trail file is of Oracle GoldenGate version 10.0 or later, which supports file headers that contain file versioning information.
- 0 means that the trail file is of an Oracle GoldenGate version that is older than 10.0. File headers are not supported in those releases. The 0 value is used for compatibility to those Oracle GoldenGate versions.

"CHARSET"
The global character set of the trail file. For example:
`WCP1252-1`

"CREATETIMESTAMP"
The time that the trail was created, in local GMT Julian time in INT64.

"URI"
The universal resource identifier of the process that created the trail file, in the format of:

host_name:dir[:dir][:dir_n] *group_name*

- `host_name` is the name of the server that hosts the process
- `dir` is a subdirectory of the Oracle GoldenGate installation path.
- `group_name` is the name of the process group that is linked with the process.

Example:

`sys1:home:oracle:v9.5:extora`
Shows where the trail was processed and by which process. This includes a history of previous runs.

"URIHISTORY"
List of the URIs of processes that wrote to the trail file before the current process.

- For a primary Extract, this field is empty.
- For a data pump, this field is `URIHistory + URI` of the input trail file.

"FILENAME"
Name of the trail file. Can be absolute or relative path, with forward or backward slash depending on the file system.

"FILEISTRAIL"
`True/false` flag indicating whether the trail file is a single file (such as one created for a batch run) or a sequentially numbered file that is part of a trail for online, continuous processing. If `false`, the `SeqNum` subtoken is not valid.

"FILESEQNO"
The sequence number of the trail file, without any leading zeros. For example, if a file sequence number is `aa000026`, `FILESEQNO` returns `26`.

"FILESIZE"
Size of the trail file. It returns `NULL` on an active file and returns a size value when the file is full and the trail rolls over.

"FIRSTRECCSN"
The commit sequence number (CSN) of the first record in the trail file.Value is `NULL` until the trail file is completed.

"LASTRECCSN"
Returns the commit sequence number (CSN) of the last record in the trail file.Value is `NULL` until the trail file is completed.

"FIRSTRECIOTIME"
The time that the first record was written to the trail file. Value is `NULL` until the trail file is completed.

"LASTRECIOTIME"
The time that the last record was written to the trail file. Value is `NULL` until the trail file is completed.

ProducerInfo: Information about the process that created the trail file

"GROUPNAME"
The group name that is associated with the Extract process that created the trail. The group name is that which was given in the `ADD EXTRACT` command. For example, "gg`ext`."

"DATASOURCE"
The data source that was read by the process. Can be one of:

- `DS_EXTRACT_TRAILS` (source was an Oracle GoldenGate extract file, populated with change data)

- `DS_LOG_TABLE` (source was an Oracle GoldenGate log table, used for trigger-based extraction)
- `DS_DATABASE` (source was a direct select from database table written to a trail, used for `SOURCEISTABLE`-driven initial load)
- `DS_TRAN_LOGS` (source was the database transaction log)
- `DS_INITIAL_DATA_LOAD` (source was Extract; data taken directly from source tables)
- `DS_VAM_EXTRACT` (source was a vendor access module)
- `DS_VAM_TWO_PHASE_COMMIT` (source was a VAM trail)

`"GGMAJORVERSION"`
The major version of the Extract process that created the trail, expressed as an integer (`xx`).

`"GGMINORVERSION"`
The minor version of the Extract process that created the trail, expressed as an integer (`xx.xx`).

`"GGMAINTENANCELEVEL"`
The maintenance version of the process (`xx.xx.xx`).

`"GGBUGFIXLEVEL"`
The patch version of the process (`xx.xx.xx.xx`).

`"GGBUILDNUMBER"`
The build number of the process.

`"GGVERSIONSTRING"`
The version string of the process. For example `11.1.1.17A not for production`.

MachineInfo: Information about the local host of the trail file

`"HOSTNAME"`
The DNS name of the computer where the Extract that wrote the trail is running. For example:

- `sysa`
- `sysb`
- `paris`
- `hq25`

`"OSVERSION"`
The major version of the operating system of the computer where the Extract that wrote the trail is running. For example:

- `Version s10_69`
- `#1 SMP Fri Feb 24 16:56:28 EST 2006`
- `5.00.2195 Service Pack 4`

"OSRELEASE"
The release version of the operating system of the computer where the Extract that wrote the trail is running. For example, release versions of the examples given for `OSVERSION` could be:

- `5.10`
- `2.6.9-34.ELsmp`
- `2000 Advanced Server`

"OSTYPE"
The type of operating system of the computer where the Extract that wrote the trail is running. For example:

- `SunOS`
- `Linux`
- `Microsoft Windows`

"HARDWARETYPE"
The type of hardware of the computer where the Extract that wrote the trail is running. For example:

- `sun4u`
- `x86_64`
- `x86`

DatabaseInfo: Information about the database that produced the data in the trail file

"DBTYPE"
The type of database that produced the data in the trail file. Some examples are:

```
DB2 UDB
DB2 ZOS
CTREE
MSSQL
MYSQL
ORACLE
SQLMX
SYBASE
TERADATA
TIMESTEN
NONSTOP
```

"DBNAME"
The name of the database, for example `findb`.

"DBINSTANCE"
The name of the database instance, if applicable to the database type, for example `ORA1022A`.

"DBCHARSET"
The character set that is used by the database that produced the data in the trail file. (For some databases, this will be empty.)

"DBMAJORVERSION"
The major version of the database that produced the data in the trail file.

"DBMINORVERSION"
The minor version of the database that produced the data in the trail file.

"DBVERSIONSTRING"
The maintenance (patch) level of the database that produced the data in the trail file.

"DBCLIENTCHARSET"
The character set of the database client.

"DBCLIENTVERSIONSTRING"
Returns the maintenance (patch) level of the database client. (For some databases, this will be empty.)

ContinuityInfo: Recovery information carried over from the previous trail file

"RECOVERYMODE"
Internal use

"LASTCOMPLETECSN"
Internal use

"LASTCOMPLETEXIDS"
Internal use

"LASTCSN"
Internal use

"LASTXiD"
Internal use

"LASTCSNTS"
Internal use

Returning Record Location and Source Application Information

Use the `RECORD` option of `@GETENV` to return location information of a record in the Oracle GoldenGate trail file or source application process information. The location information uniquely identifies a record through the sequence number of the trail file and the relative byte address or the transaction identifier. Source application information identifies the source program that alters the Enscribe file record.

Syntax

```
@GETENV ("RECORD", "option")
```

"`option`" is one of options described in the following sections:

- "Record Location Options"
- "Source Application Options"

Record Location Options

FILERBA
Returns the relative byte address (RBA) of the record within the `FILESEQNO` trail file.

FILESEQNO
Returns the sequence number of the trail file without any leading zeros.

TRANSID
Returns the TMF transaction identifier for which the record was altered.

Source Application Options

PROGRAMNAME
Returns the name of the source application program that altered the Enscribe file record.

PROCESSNAME
Returns the process identifier (PID) of the source application process that altered the Enscribe file record.

Associating BASE24 Keys and Records

Use the `TLFKEY` option of `@GETENV` to associate a unique key with `TLF/PTLF` records in the ACI BASE24 application. The 64-bit key is composed of the following concatenated items:

- the number of seconds since `2000`.
- the block number of the record in the `TLF/PTLF` block multiplied by ten.
- the node specified by the user (must be between `0` and `255`).

This option is valid for the Extract and Replicat processes.

Syntax

```
@GETENV ("TLFKEY", SYSKEY, unique_key)
```

unique_key
The NonStop node number of the source `TLF/PTLF` file.
Example: `@GETENV ("TLFKEY", SYSKEY, 7)`

GETVAL

Use the `@GETVAL` function to extract values from a query so that they can be used as input to a `FILTER` or `COLMAP` clause of a `MAP` or `TABLE` statement.

Whether or not a parameter value can be extracted with `@GETVAL` depends upon the following:

1. Whether or not the query executed successfully.
2. Whether or not the query results have expired.

Handling Missing Column Values

When a value cannot be extracted, the `@GETVAL` function results in a "column missing" condition. Typically, this occurs for update operations if the database only logs values for columns that were changed.

Usually this means that the column cannot be mapped. To test for missing column values, use the `@COLTEST` function to test the result of `@GETVAL`, and then map an alternative value for the column to compensate for missing values, if desired. Or, to

ensure that column values are available, you can use the FETCHCOLS or FETCHCOLSEXCEPT option of the TABLE or MAP parameter to fetch the values from the database if they are not present in the log. (Enabling supplemental logging for the necessary columns also would work.)

Syntax

```
@GETVAL (name.parameter)
```

name
The name of the query. When using SQLEXEC to execute the query, the valid value is the logical name specified with the ID option of the SQLEXEC clause. ID is a required SQLEXEC argument for queries.

parameter
Valid values are one of the following.

- The name of the parameter in the query from which the data will be extracted and passed to the column map.
- RETURN_VALUE, if extracting values returned by a query.

Example

The following example enables the COMPUTE statements to call two stored procedures selectbal and selecttran by referencing the logical name within the @GETVAL function and referring appropriately to the results of each.

```
MAP \NY.$DATA1.GGSDAT.ACCTTR, TARGET \NY.$DATA1.GGSDAT.ACCTBL
SQLEXEC (ID selecttran, ON UPDATES, ON INSERTS,
QUERY " select tran_type, tran_amt from $DATA1.GGSDAT.ACCTTR "
" where ACCT_NUM = ?P1 ",
PARAMS (P1 = ACCT_NUM), ERROR REPORT)

SQLEXEC (ID selectbal, ON UPDATES, ON INSERTS,
QUERY " select acct_balance from $DATA1.GGSDAT.ACCTBL "
" where ACCT_NUM = ?P1 ",
PARAMS (P1 = ACCT_NUM), ERROR REPORT)

COLMAP (USEDEFAULTS,
acct_balance =
@IF (@GETVAL (selecttran.tran_type) = 1
@COMPUTE (@GETVAL (selectbal.acct_balance) - selecttran.tran_amt),
@COMPUTE (@GETVAL (selectbal.acct_balance) + selecttran.tran_amt))
);
```

HIGHVAL | LOWVAL

Use the @HIGHVAL and @LOWVAL functions when you need to generate a value, but you want to constrain it within an upper or lower limit. These functions emulate the COBOL functions of the same name.

Use @HIGHVAL and @LOWVAL only with string and binary data types. Using them with decimal or date data types, or with SQLEXEC, can cause errors.

> **Note:**
>
> Invalid maps to incorrect type will result in a mapping error 222.

Syntax

```
@HIGHVAL ([length]) | @LOWVAL ([length])
```

length
Optional. Specifies the binary output length in bytes. The maximum value of *length* is the length of the target column.

Examples

Example 1
This example sets COBOL-type group level to low values if key is less than `50`, and it sets COBOL-type group level to high values if the key is greater than `50`.

```
MAP \PROD.$DATA.MASTER.CUSTOMER, TARGET \BACK.$DATA.MASTER.CUSTOMER, DEF CUSTOMER-
REC,
TARGETDEF NEW_CUSTOMER_REC,
COLMAP (USEDEFAULTS, CUST-KEY = CUST-KEY,
        GROUP-LEVEL = @IF (CUST-KEY < 50,@LOWVAL(), @HIGHVAL()));
```

Example 2
The following example assumes that the size of the `GROUP-LEVEL` field is 5 bytes.

Function statement	Results
`GROUP-LEVEL = @HIGHVAL ()`	`{0xFF, 0xFF, 0xFF, 0xFF, 0xFF}`
`GROUP-LEVEL = @LOWVAL ()`	`{0x00, 0x00, 0x00, 0x00, 0x00}`
`GROUP-LEVEL = @HIGHVAL (3)`	`{0xFF, 0xFF, 0xFF}`
`GROUP-LEVEL = @LOWVAL (3)`	`{0x00, 0x00, 0x00}`

IF

`@IF` returns one of two values, based upon a condition.

Syntax

```
@IF (conditional_expression, nonzero_value, zero_value)
```

conditional_expression
The conditional expression.

non-zero_value
The value if the expression is non-zero. A non-zero result is considered `TRUE`.

zero_value
The value if the expression is zero. A zero result is considered `FALSE`.

Examples

Example 1
The following returns `AMT` only if `AMT` is greater than zero, otherwise zero is returned.

```
AMOUNT_COL = @IF (AMT <= 0, 0, AMT)
```

Example 2
The following returns `WEST` if `STATE` is `CA`, `AZ` or `NV`, otherwise returns `EAST`.

```
REGION = @IF (@VALONEOF (STATE, "CA", "AZ", "NV"), "WEST", "EAST")
```

Example 3
The following returns `NULL` unless both `PRICE` and `QUANTITY` are greater than zero.

```
ORDER_TOTAL = @IF (PRICE > 0 AND QUANTITY > 0, PRICE * QUANTITY, @COLSTAT(NULL)
```

Example 4
The following returns `NULL` unless both `PRICE` and `QUANTITY` are greater than zero. `COLSTAT(NULL)` creates a null value in the target column.

```
ORDER_TOTAL = @IF (PRICE > 0 AND QUANTITY > 0, PRICE * QUANTITY,
    @COLSTAT(NULL)
```

Example 5
The following returns `NULL` if either `PRICE` or `QUANTITY` is `NULL`. When any columns in an expression are `NULL`, this is the default action.

```
ORDER_TOTAL = @IF (@COLTEST (PRICE, NULL) OR @COLTEST(QUANTITY, NULL),
@COLSTAT(NULL), PRICE * QUANTITY)
```

NUMBIN

`@NUMBIN` turns a binary string of eight or fewer bytes, into a number. Use this when the source DDL defines a byte stream that is actually a number as a string.

Syntax

```
@NUMBIN (source_column)
```

NUMSTR

`@NUMSTR` converts a string (character) field or value into a number. Use this to map a string field into a number, or to use a string field that contains only numbers in an arithmetic expression.

Syntax

```
@NUMSTR (convert_field)
```

convert_field
A character column or a literal string.

RANGE

The `@RANGE` function, used within the `FILTER` option, helps divide workload into multiple, randomly distributed groups of data, while guaranteeing that the same row will always be processed by the same program. For example, `@RANGE` can be used to split the workload by different key ranges for a heavily accessed table into different Replicat processes.

The user specifies both a range that applies to the current process, and the total number of ranges (generally the number of processes), and optionally a list of column names to use to calculate the range against.

`@RANGE` computes a hash value of all the columns specified, or if no columns are specified, the primary key columns of the source table. A remainder of the hash and the total number of ranges is compared with the ownership range to determine whether or not `@RANGE` produces true or false results. Oracle GoldenGate adjusts the total number of ranges so that they are evenly distributed.

> **Note:**
>
> Calculating ranges in an Extract parameter file is more efficient that doing so in a Replicat parameter file. Calculating ranges on the target requires Replicat to read all of the Oracle GoldenGate trail data to find the data meeting each range specification.

> **Note:**
>
> Using the `@RANGE` function within a `FILTER` provides different capabilities, such as specifying columns, than using the `RANGE` option of `FILE` or `MAP`. And both of these are different than the `RANGE` option of `ALTINPUT`.

Syntax

```
@RANGE (range, total_ranges [, column] [, column] [, ...]))
```

range
The range assigned to the specified process or trail. Valid values are `1`, `2`, `3`, and so forth, with the maximum value being the value defined by *`total_ranges`*.

total_ranges
The total number of ranges allocated. For example, to divide data into three groups, use the value `3`.

column
The name of a column, or columns, on which to base the range allocation. This argument is optional. If not used, Oracle GoldenGate calculates ranges based on the table's primary key.

Examples

Example 1
In the following example, the workload is split into three ranges, between three Replicat processes, based on the `ID` column of the `SRCTAB` table.
Replicat parameter file #1 contains:

```
MAP $PRODSRC.PRODMSTR.SRCTAB, TARGET $PROD.MASTER.TARGTAB,
FILTER (@RANGE(1,3, ID));
```

Replicat parameter file #2 contains:

```
MAP $PRODSRC.PRODMSTR.SRCTAB, TARGET $PROD.MASTER.TARGTAB,
FILTER (@RANGE(2,3, ID);
```

Replicat parameter file #3 contains:

```
MAP $PRODSRC.PRODMSTR.SRCTAB, TARGET $PROD.MASTER.TARGTAB, FILTER (@RANGE(3,3,
ID));
```

Example 2
In the following example, the `TABLE` parameter in the Extract parameter file splits the processing load into two trails. Since no columns were defined on which to base the range calculation, Oracle GoldenGate will use the primary key columns.

```
RMTTRAIL $DATA.GGSDAT.AA
TABLE ACCOUNT, FILTER (@RANGE (1, 2));
RMTTRAIL $DATA.GGSDAT.BB
TABLE ACCOUNT, FILTER (@RANGE (2, 2));
```

Example 3
In the following example, the `ORDMASTR` table has a key of `ORDERID` and the `ORDDETL` table has a key of `ITEMNUM`. Because the key `ORDERID` establishes relativity, it is used in `@RANGE` filters for both tables to preserve referential integrity. The load is split into two ranges.
(Parameter file #1)

```
MAP $PRODSRC.PRODMSTR.ORDMASTR, TARGET $PROD.MASTER.ORDMASTR,
FILTER (@RANGE (1, 2, ORDERID));
MAP $PRODSRC.PRODMSTR.ORDDETL, TARGET $PROD.MASTER.ORDDETL,
FILTER (@RANGE (1, 2, ORDERID));
```

(Parameter file #2)

```
MAP $PRODSRC.PRODMSTR.ORDMASTR, TARGET $PROD.MASTER.ORDMASTR,
FILTER (@RANGE (2, 2, ORDERID));
MAP $PRODSRC.PRODMSTR.ORDDETL, TARGET $PROD.MASTER.ORDDETL,
FILTER (@RANGE (2, 2, ORDERID));
```

STRCAT

Use the `@STRCAT` function to concatenate one or more strings. The string can be either the name of a column or a literal string. Enclose literals within quotes.

Syntax

```
result = @STRCAT (string1, string2 [,...])
```

string1
The first string to be concatenated.

string2
The second string to be concatenated.

Example

The following creates a phone number from three fields and includes the constant values.

```
PHONE_NO = @STRCAT ("(", AREA_CODE,")",PREFIX,"-", PHONE)
```

STRCMP

Use the `@STRCMP` function to compare two character columns or literal strings. Enclose literals within quotes.

`@STRCMP` returns the following:

- `-1` if the first string is less than the second.
- `0` if the strings are equal.
- `1` if the first string is greater than the second.

Trailing spaces are truncated before comparing the strings.

Syntax

```
@STRCMP (string1, string2)
```

string1
The first column or literal string to be compared.

string2
The second column or literal string to be compared.

Example

The following example compares two literal strings and returns `1` because the first string is greater than the second.

```
@STRCMP ("JOHNSON", "JONES")
```

STREQ

Use `@STREQ` to determine if two strings are equal. The result is either:

- `0` = strings are not equal
- `1` = strings are equal

Syntax

```
result = @STREQ (string1, string2)
```

string1
The first string to compare.

string2
The second string to compare.

Example

The following filter clause compares the value of the variable `REGION` to the literal value `"EAST"`. If `Region = EAST`, the record passes the filter.

```
FILTER (@STREQ (REGION, "EAST"))
```

STREXT

Use the `@STREXT` function to extract a portion of a string.

Syntax

```
result = @STREXT (string, begin_position, end_position)
```

string
The string from which to extract. The string can be either the name of a column or a literal string. Enclose literals within quotes.

begin_position
The character position at which to begin extracting.

end_position
The character position at which to end extracting. The end position is included in the extraction.

Example

The following example uses three `@STREXT` functions to extract a phone number into three different columns.

```
AREA_CODE = @STREXT (PHONE, 1, 3),
PREFIX    = @STREXT (PHONE, 4, 6),
PHONE_NO  = @STREXT (PHONE, 7, 10)
```

STRFIND

Use the `@STRFIND` function to determine the position of a string within a string column or else return zero if not found. Optionally, `@STRFIND` can accept a starting position to search within the string.

Syntax

```
result = @STRFIND (string, "search_string" [, begin_position])
```

string
The string from which to extract. The string can be either the name of a column or a literal string. Enclose literals within quotes.

"search_string"
The string for which to search within the string. Enclose the search string within quotes.

begin_position
The character position at which to begin searching.

Example

Assuming the string for `ACCT` is `ABC123ABC`, the following are possible results.

```
result = @STRFIND (ACCT, "23") returns 5.
result = @STRFIND (ACCT, "ZZ") returns 0.
result = @STRFIND (ACCT, "ABC", 2) returns 7.
```

STRLEN

Use the `@STRLEN` function to return the length of a string, in number of characters.

Syntax

```
result = @STRLEN (string)
```

string
Can be the name of a column or a literal string. Enclose literals within quotation marks.

Example

```
@STRLEN (ID_NO)
```

STRLTRIM

Use the `@STRLTRIM` function to trim leading spaces.

Syntax

```
@STRLTRIM (string)
```

string
Can be the name of a character column or a literal string. Enclose literals within quotes.

Example

```
  birth_state = @strltrim(state)
```

STRNCAT

Use the `@STRNCAT` function to concatenate one or more strings to a maximum length.

Syntax

```
result = @STRNCAT (string, max_length [, string, max_length, ...])
```

string
Can be the name of a column or a literal string. Enclose literals within quotation marks.

max_length
The maximum string length, in characters.

Example

The following concatenates two strings and results in "ABC123."

```
PHONE_NO = @STRNCAT ("ABCDEF", 3, "123", 3)
```

STRNCMP

Use @STRNCMP to compare two strings, up to a specified number of characters in each string. Trailing spaces are truncated before comparing the strings. The compare returns:

- 0 if the strings are equal.
- -1 if the first string is less than the second string.
- 1 if the first string is greater than the second string.

Syntax

```
@STRNCMP (compare_data, compare_data, max_compare_length)
```

compare_data
The data to compare. Can be a character column or literal string.

max_compare_length
Specifies a number of characters to be compared in each string. For example, if you specify 2, the first two characters of each string are compared. If they are equal, 0 (zero) is returned.

Example

This example returns 0, since the first two characters of both strings are equal.

```
result = @STRNCMP ("JOHNSON", "JONES", 2)
```

STRNUM

Use @STRNUM to convert a number into a string and specify the output format and padding.

Syntax

```
@STRNUM (field, {LEFT | LEFTSPACE | RIGHT | RIGHTZERO} [length])
```

field
The name of the source numeric field.

LEFT
Left justify, fill the rest of the target column with spaces

LEFTSPACE
Left justify, fill the rest of the target column.

RIGHT
Right justify, fill with spaces

RIGHTZERO
Right justify, fill the rest of the target column with zeros

length
Specifies the output length, when any of the options are used that specify padding (all but `LEFT`).

Example

If field `NUM` has the value 15 and the target column `CHAR1` is a maximum of 5 characters, the following examples show the different types of results obtained with formatting options.

Function statement	Results (- denotes a space)
`CHAR1 = @STRNUM (NUM, LEFT)`	`15`
`CHAR1 = @STRNUM (NUM, LEFTSPACE)`	`15---`
`CHAR1 = @STRNUM (NUM, RIGHTZERO)`	`00015`
`CHAR1 = @STRNUM (NUM, RIGHT)`	`---15`

If an output *length* of 4 is specified in the preceding example, the following shows the different types of results.

Function statement	Results (- denotes a space)
`CHAR1 = @STRNUM (NUM, LEFTSPACE, 4)`	`15--`
`CHAR1 = @STRNUM (NUM, RIGHTZERO, 4)`	`0015`
`CHAR1 = @STRNUM (NUM, RIGHT, 4)`	`--15`

STRRTRIM

Use the `@STRRTRIM` function to trim trailing spaces.

Syntax

`@STRRTRIM (`*string*`)`

string
Can be the name of a character column or a literal string. Enclose literals within quotes.

Example

```
street_address = @strrtrim(address)
```

STRSUB

Use `STRSUB` to substitute one string within another string field or constant.

Syntax

```
@STRSUB (source_string_or_col, search_string, substitute_string, ...)
```

source_string_or_col
The string or column to replace.

search_string
The value to be replaced.

substitute_string
The replacement value.

Examples

Example 1
For this example, the source string is `"123ABC123"`. The value `"123"` is to be replaced with `"xx"`. The result is `xxABCxx`.

```
result = @STRSUB ("123ABC123", "123", "xx")
```

Example 2
For this example, the source string is "`123ABC123`". The value "`A`" is to be replaced with "`z`" and the value `"1"` is replaced with `"0"`. The result is `023zBC023`.

```
result = @STRSUB ("123ABC123", "A", "z", "1", "0")
```

STRTRIM

Use the `@STRTRIM` function to trim leading and trailing spaces.

Syntax

```
@STRTRIM (string)
```

string
Can be the name of a character column or a literal string. Enclose literals within quotes.

Example

```
 pin_no = @strtrim(custpin)
```

STRUP

Use `@STRUP` to change a character string or field to uppercase.

Syntax

```
@STRUP (string)
```

string
Can be the name of a character field or a literal string. Enclose literals within quotes.

Example

The following changes the string, "aaaaa" to "AAAAA".

```
result = @STRUP ("aaaaa")
```

TOKEN

Use the @TOKEN function to retrieve data that is stored in the user token area of the Oracle GoldenGate trail record header. Tokens are defined in the Extract parameter file by means of the TOKENS clause of the FILE or TABLE parameter. The token data can be mapped to a target column by means of a COLMAP clause or used within a SQLEXEC statement or Oracle GoldenGate macro or user exit.

Syntax

```
@TOKEN ("token_name")
```

"token_name"
The name of the token for which data is to be retrieved.

VALONEOF

Use @VALONEOF to compare a field or string to a list of values. If the field is in the list, 1 is returned, otherwise 0 is returned.

Syntax

```
@VALONEOF (expression, value [, value] [, ...])
```

Example

If STATE is CA or NY, this expression returns "COAST".

```
@IF (@VALONEOF (STATE, "CA", "NY"), "COAST", "MIDDLE")
```

COLSTAT

COLSTAT returns an indicator to the Extract and Replicat programs that a field is MISSING, NULL, or INVALID.

Syntax

```
@COLSTAT (MISSING | INVALID | NULL)
```

Example

The following example returns a NULL into target column DATE1.

```
DATE1 = @COLSTAT (NULL)
```

DATENOW

`@DATENOW` takes no arguments and returns the current date and time in the format `YYYY-MM-DD HH:MI:SS.FFFFFF`. The date and time are returned in local time, including adjustments for daylight savings time.

Syntax

```
@DATENOW ()
```

5
DEFGEN Arguments

Learn how to use the runtime arguments for the DEFGEN utility. These arguments are entered at the command line when you run the utility. The DEFGEN utility produces a file defining the layouts of the source files and tables. The definitions are used by Collector and Replicat and, in some cases, Extract.
For more information about using DEFGEN, see Generating Data Definitions with DEFGEN.

Argument Summary

DEFGEN accepts the following arguments.

Argument	Description
EXCLUDESYSTEM | INCLUDESYSTEM	Instructs DEFGEN to omit (EXCLUDESYSTEM) or include (INCLUDESYSTEM) the NonStop system name from the files and tables for which definitions are being generated.
EXPANDDDL	Supplies options to manipulate output for Enscribe record definitions.
NCHARCOMPATIBILITY	Sets the datatype to 0 from 2 for multibyte nchar and 64 from 66 for multibyte nvarchar.
RECORDNAMEPROMPTING	Supplies the name of an existing record definition to use when generating a definition for a new table.
FORMAT LEVEL	This is only to be used when the definition file produced will be used for the TARGETDEFS of an OpenSys Extract to map from another database to NSK.

EXCLUDESYSTEM | INCLUDESYSTEM

Use EXCLUDESYSTEM and INCLUDESYSTEM to omit or include the NonStop system name from the files and tables for which definitions are being generated. Must precede the EXPANDDDL argument.

Default

```
INCLUDESYSTEM
```

Syntax

```
EXCLUDESYSTEM | INCLUDESYSTEM
```

EXPANDDDL

Use EXPANDDDL options 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. If used, it must be the last argument entered.

Syntax

```
EXPANDDDL separator option [, ...]
```

separator
A character separator for defining array output. See "Defining a separator ".

option
For information about the options, see:

- Information on inheriting OCCURS on "Inheriting OCCURS ".
- "Resolving duplicate field names ".
- "Omitting or including redefined fields ".
- "Fixing long field names ".
- "Altering field array display ".

Defining a separator

Use separators for defining array output into columns. If any option besides NOEXPANSION is specified, a distinct field is output for each occurrence of a given field and a grouping field with the original name.

For example, consider the DDL definition:

```
03 FIELDX PIC 9(5) OCCURS 2 TIMES.
```

Normally, the array is output as a single field with three occurrences. However, arrays do not exist in an SQL environment, so it may be desirable to create a column for each occurrence of FIELDX. If the USEDASH option is specified, the following fields are output:

```
FIELDX
FIELDX-1
FIELDX-2
```

FIELDX references all occurrences as a group item, while -1 and -2 are the individual occurrences.

If you are specifying a separator it must be the first option in the command string.

Default

```
USEDASH
```

Syntax

```
EXPANDDDL separator
```

separator is one of the following.

USEBRACKETS
Use brackets [ ].

USEDASH
Use a dash -.

USEUNDERSCORE
Use an underscore _.

USETWOUNDERSCORES
Use two underscores __.

NOEXPANSION
Do not use a separator.

Inheriting OCCURS

Use the `EXPANDGROUPARRAYS` option to append indexes to fields that do not necessarily occur multiple times, but which are part of groups that occur multiple times.

Default

`NOEXPANDGROUPARRAYS`

Syntax

```
EXPANDDDL USEUNDERSCORE EXPANDGROUPARRAYS
```

Example

For the following definition:

```
03 A OCCURS 2 TIMES.
   04 B PIC 9(5).
03 X OCCURS 4 TIMES.
   04 Y PIC 9(5) OCCURS 4 TIMES.
```

`B` within the second occurrence of `A` becomes `B_2`. The third occurrence of `Y` within the fourth occurrence of `X` becomes `Y_4_3`.

Resolving duplicate field names

You can resolve duplicate file names by specifying a duplicate field option for:

- Unique numerical index to the end of a duplicate field or column.
- Unique alpha character index to the end of a duplicate field or column.
- Group name at the beginning of a duplicate field or column.

Default

`NORESOLVEDUPS`

Syntax

`EXPANDDDL` *`dup_field_option`*

`dup_field_option` is one of the following.

RESOLVEDUPINDEX
Appends a numerical index to the end of each duplicate field or column. For example, `END-DATE.YY` would become `YY-2` in the output since it is the second occurrence of `YY` in the definition.

RESOLVEDUPALPHAINDEX
Appends an alpha character index to the end of redundant fields or columns; `BEGIN-DATE.MM` would become `MM-A` (occurrences are designated `A-Z`).

RESOLVEDUPGROUP
Prefixes a duplicate field name with its group name and separator.

RESOLVEDUPFULLNAME
Use only when `RESOLVEDUPGROUP` cannot resolve duplicates. `RESOLVEDUPFULLNAME` triggers the system to fully qualify the field name so that the redundancy is resolved.

NORESOLVEDUPS
Do not resolve duplicate field names.

Examples

Example 1
Consider the following definition.

```
03 BEGIN-DATE.
04 YY    PIC 99.
04 MM    PIC 99.
04 DD    PIC 99.
03 END-DATE.
04 YY    PIC 99.
04 MM    PIC 99.
04 DD    PIC 99.
```

`BEGIN-DATE` and `END-DATE` duplicate the `YY`, `MM` and `DD` field names. Normally these duplications would be resolved by referencing the higher level group item, such as `BEGIN-DATE.DD` or `END-DATE.DD`. The `RESOLVEDUPGROUP` duplicate field option prefixes each duplicate field name with its group name, as in `BEGIN-DATE-DD` and `END-DATE-DD`.

Example 2
In this example, the first non-redundant group above the elementary item is used for the prefix. For example, `END-DATE.DATE-DEF.YY` would become `END-DATE-YY` (`DATE-DEF` is omitted).

```
03 BEGIN-DATE.
04 DATE-DEF.
05 YY    PIC 99.
05 MM    PIC 99.
05 DD    PIC 99.
03 END-DATE.
04 DATE-DEF.
05 YY    PIC 99.
05 MM    PIC 99.
05 DD    PIC 99.
```

Omitting or including redefined fields

Use `OMITREDEFS` or `INCLUDEREDEFS` to omit or include redefined fields in the output definition. Including redefinitions has the following consequences.

- DDLGEN outputs a column for every elementary field item in the source definition, regardless of whether or not it redefines another field.
- When the Extract program specifies FORMATASCII or FORMATSQL, the Collector outputs an ASCII field for each redefined field.

Default

OMITREDEFS

Syntax

```
EXPANDDDL [OMITREDEFS | INCLUDEREDEFS]
```

OMITREDEFS
Omit redefined fields.

INCLUDEREDEFS
Include redefined fields.

Fixing long field names

Use MAXCOLNAMELEN to manage long field names. By default, the maximum field name length is 30. You can change this value with the MAXCOLNAMELEN option.

When creating new field names to resolve duplicate occurrences, names occasionally exceed the length specified by MAXCOLNAMELEN. When this happens, you are prompted for an alternative column name. To avoid the prompt, specify the NOFIXLONGNAMES option after MAXCOLNAMELEN.

MAXCOLNAMELEN is invoked only with EXPANDGROUPARRAYS or RESOLVEDUPGROUP.

Default

FIXLONGNAMES (prompt for an alternative column name)

Syntax

```
EXPANDDDL [MAXCOLNAMELEN length | [NOFIXLONGNAMES]]
```

length
The field name length.

NOFIXLONGNAMES
Specify to prevent prompting for a new column name. A warning message is issued.

Altering field array display

Use ZEROFILL to expand output definitions to a constant width, similar to:

```
EXPANDDDL USEUNDERSCORE ZEROFILL 3
EXPANDDDL USEUNDERSCORE ZEROFILL ARRAYWIDTH
```

Default

ZEROFILL 1

Syntax

```
EXPANDDDL {ZEROFILL width} | {ARRAYWIDTH}
```

width
A value indicating the character width of the output.

ARRAYWIDTH
Specifies that the output width match the number of elements in the array.

Example 1
If ZEROFILL ARRAYWIDTH is specified for definition 03 FLAG PIC X OCCURS 500 TIMES, then occurrence 7 becomes FLAG-007 and occurrence 423 becomes FLAG-423. The width is equal to the occurrence frequency (500) and is three characters wide.

Example 2
Using the definition from the previous example and specifying ZEROFILL 5 results in occurrence 7 becoming FLAG-00007 and occurrence 423 becoming FLAG-00423.

NCHARCOMPATIBILITY

Use NCHARCOMPATIBILITY to cause DEFGEN to specify single byte data types for columns that are multibyte. The data type is changed from 2 to 0 for multibyte nchar columns and from 66 to 64 for multibyte nvarchar columns. Must precede the EXPANDDDL argument.

Default

No change to the datatypes

Syntax

```
NCHARCOMPATIBILITY
```

RECORDNAMEPROMPTING

Use RECORDNAMEPROMPTING to trigger DEFGEN to prompt for the name of an existing record definition instead of a file. Use this parameter to enter the record name when the same definition is to be used for multiple tables with identical definitions made up of the same columns, column order, and data types. Must precede the EXPANDDDL argument.

Default

File name prompting

Syntax

```
RECORDNAMEPROMPTING
```

FORMAT LEVEL

This should be only used when the definition file produced will be used for the TARGETDEFS of an OpenSys Extract to map from another database to NSK. The release number should be the release of the OpenSys Extract that is used . Must precede the EXPANDDDL argument.

0 = Releases prior to 10.0
1 = 10.0
2 = 10.4

3 = 11.2
4 = 12.1
5 = 12.2

Default

```
0
```

Syntax

```
FORMAT LEVEL 5
```

6
DDLGEN Arguments

Learn how to use the runtime arguments for the DDLGEN utility. These arguments are entered at the command line when you run the utility. DDLGEN utility provides instructions for generating table definitions for target databases based on existing Enscribe and SQL definitions.
For more information about using DDLGEN, see Creating Target Database DDL.

Run-time arguments

Syntax

```
TACL> RUN DDLGEN /IN [command_file]/ [-d file_name]
```

The DDLGEN syntax shown above includes the following run-time arguments:

/IN command_file/
If you have created and saved a file of your responses using the NonStop editor, you can enter the name of that file.

-d file_name
Instructs DDLGEN to use the DEFGEN definitions file *file_name*.

OMITNULL | INCLUDENULL

Use OMITNULL and INCLUDENULL to omit or include the word NULL for columns that can contain null values. Assuming the column can be set to null, DDLGEN outputs the following definitions:

- If INCLUDENULL (or nothing) is specified, the output is:

  ```
  COL CHAR (10) NULL
  ```

- If OMITNULL is specified, the output is:

  ```
  COL CHAR (10)
  ```

The default templates for each supported database already include the correct setting for this argument.

Default

```
INCLUDENULL
```

Syntax

```
OMITNULL | INCLUDENULL
```

OMITNOTNULL | INCLUDENOTNULL

Use `OMITNOTNULL` and `INCLUDENOTNULL` to omit or include the words `NOT NULL` for columns that cannot contain null values. Assuming the column cannot be set to null, `DDLGEN` outputs the following definitions:

- If `INCLUDENOTNULL` (or nothing) is specified:

  ```
  COL CHAR (10) NOT NULL
  ```

- If `OMITNOTNULL` is specified:

  ```
  COL CHAR (10)
  ```

The default templates for each supported database already include the correct setting for this argument.

Default

```
INCLUDENOTNULL
```

Syntax

```
OMITNOTNULL | INCLUDENOTNULL
```

OMITREDEFS | INCLUDEREDEFS

Use `OMITREDEFS` and `INCLUDEREDEFS` to omit or include fields that redefine other fields in a source DDL definition in the output table definition.

Default

```
INCLUDEREDEFS
```

Syntax

```
OMITREDEFS | INCLUDREDEFS
```

OMITCOMMENTS | INCLUDECOMMENTS

Use `OMITCOMMENTS` and `INCLUDECOMMENTS` to omit or include comments regarding anomalies such as duplicate field names, arrays and other items that were transformed or need to be corrected manually.

Default

```
INCLUDECOMMENTS
```

Syntax

```
OMITCOMMENTS | INCLUDECOMMENTS
```

USESAMENULLS

Use `USESAMENULLS` to use the same `NULL` attribute as the source column.

Default

Do not use the same NULL attributes.

Syntax

USESAMENULLS

Template parameters

Oracle GoldenGate includes a template for each supported database. The following arguments can be used in these templates. For more information about using templates, see Understanding the Template File.

Parameters	Description
OMITNULL | INCLUDENULLOMITNULL | INCLUDENULL	Determines whether or not columns contain NULL.
OMITNOTNULL | INCLUDENOTNULL	Omits or includes NOT NULL from applicable columns.
OMITREDEFS | INCLUDEREDEFS	Omits or includes fields that are redefines of other fields.
OMITCOMMENTS | INCLUDECOMMENTS	Omits or includes comments.
USESAMENULLS	Directs DDLGEN to use the same NULL attribute as the source column.

7

User Exit Functions

Learn how to use the user exits to extend and customize the functionality of Oracle GoldenGate for HP NonStop on Guardian.
This chapter includes the following sections:

User exits extend and customize the functionality of the Extract and Replicat processing by calling COBOL, TAL or C routines from the Extract or Replicat parameter files. At different points during processing you can invoke custom COBOL, TAL or C routines to perform specialized functions.

Reference Guide for HP NonStop (Guardian) provide:

- Function parameters for accepting and processing different events and information from Extract or Replicat.
- Calling environment functions, in TAL, C and COBOL, for retrieving context information from Extract or Replicat.

Two example user exits are supplied with the installation code: `DEMOXCOB` (written in COBOL) and `DEMOXC` (written in C). You can use these exits as a sample to help plan your own user exit routines. Two blank user exit templates are also included: `USEREXC` for `C` and `USEREXT` for `TAL`. You can use these to start coding your routines. There is no sample for `TAL` and no template for `COBOL`.

> **Note:**
>
> User exits may be used for many kinds of routines; however, you should avoid controlling transaction commits with custom user exit logic. This could interfere with Oracle GoldenGate checkpoint processing, which facilitates troubleshooting and protects you in case of system failure.

Function summary

Function parameters accept different events and information from the Extract or Replicat program, pass the information to the appropriate routine for processing and return a response and information to the caller.

Routine	Description
EXIT_CALL_RESULT	Set by the user exit routines and tells the caller how to respond when each exit call completes.
EXIT_CALL_TYPE	Indicates the processing point of the caller and determines the type of processing to perform.
EXIT_PARAMS	Supplies information about the associated EXIT_REC_BUF when the call type is PROCESS_RECORD or DISCARD_RECORD. The exit routines can alter certain EXIT_PARAMS to change how the caller processes returned records.
EXIT_REC_BUF	Contains the record about to be processed by Extract or Replicat. EXIT_REC_BUF is supplied when the call type is PROCESS_RECORD or DISCARD_RECORD. Exit routines can change the contents of this buffer, for example, to perform custom mapping functions. When changing the contents of EXIT_REC_BUF, change the SOURCE_FILE (for Extract processing only) and RECORD_LEN parameters appropriately so that external processes can identify the record.

EXIT_CALL_RESULT

EXIT_CALL_RESULT is set by the user exit routines and tells the caller how to respond when each exit call completes. The following results can be specified by the operator's routines.

Result	Description
EXIT_ABEND_VAL	Instructs the caller to ABEND immediately.
EXIT_IGNORE_VAL	Reject records for further processing. IGNORE is appropriate when the exit performs all the desired processing on a record and there is no desire to output or replicate it.
EXIT_OK_VAL	If the routine does nothing to respond to an event, OK is assumed. If the call specified PROCESS-RECORD or DISCARD-RECORD and OK is returned, the caller processes the record buffer returned by the user exit and uses the parameters set by the exit (see the EXIT-REC-BUF parameter for details).
EXIT_STOP_VAL	Instructs the caller to STOP immediately. STOP or ABEND may be appropriate when an error condition occurs in your program, such as a file error.

EXIT_CALL_TYPE

EXIT_CALL_TYPE indicates the processing point of the caller and determines the type of processing to perform. Extract and Replicat use the following types of calls.

Call	Description
EXIT_CALL_BEGIN_TRANS	In Extract, invoked just before the output of the first record in a transaction. In Replicat, invoked just before the start of a replicated transaction.

Call	Description
EXIT_CALL_DISCARD_RECORD	Called when a record is discarded. Records can be discarded for several reasons, such as the changed record is out-of-sync with the current version of an SQL table. When DISCARD-RECORD is specified, the associated buffer is passed and custom discard processing can be specified.
EXIT_CALL_END_TRANS	In Extract and Replicat, invoked just after the last record in a transaction is processed.
EXIT_CALL_FILE_CLOSE	Invoked by a direct read Extract when it closes the current data file.
EXIT_CALL_PROCESS_RECORD	In Extract, invoked before a record buffer is output to a trail or file. In Replicat, invoked just before a Replicat operation is performed. This call is the basis of most user exit processing. When a PROCESS_RECORD call is invoked, the record buffer and parameters describing the record are supplied to the user exit. You can map, transform, clean or perform virtually any other operation with the record. You can return a status indicating whether the caller should process or ignore the record.
EXIT_CALL_START	Invoked at the start of processing. The user exit can perform initialization work, such as opening files, initializing variables, and so forth.
EXIT_CALL_STOP	Invoked before the caller stops or ends abnormally. The user exit can perform completion work, such as closing files or outputting totals.

EXIT_PARAMS

EXIT_PARAMS supply information about the associated EXIT-REC-BUF when the call type is PROCESS_RECORD or DISCARD_RECORD. The exit routines can alter certain EXIT_PARAMS to change how the caller processes returned records. The EXIT_PARAMS structure includes the following:

Exit Parameter	Description
AUDIT_TIMESTAMP	Provides the Julian, GMT timestamp of the audit block in which the record was found. This timestamp approximates the time of the original database operation.
BEFORE_AFTER_IND	For update records, determines whether the record is a before-image (B) or after-image (A). When extracted, before-images precede after-images within the same update. Inserts are after-images, and deletes are before-images.
ERR_INFO	This information is supplied with a discarded record to indicate the reason for the discard. The error number (ERR_NUM) is passed, along with a possible file or SQL error and error text.
FUNCTION_PARAM	Lets you pass a parameter to the exit. This parameter is set up in the Extract or Replicat parameter file as part of the FILE or MAP parameter. The default size of EXIT_PARAMS is 256 bytes. If the string supplied with the EXITPARAM parameter is shorter, it will be a NULL terminated string. Use the GET_EXITPARAM_VALUE function to access data over that length.

Exit Parameter	Description
MORE_RECS_IND	Set by the user on return from an exit. For database records, determines whether Extract or Replicat processes the record again. This allows the exit program, for example, to output many records per record processed by Extract, a common function when converting Enscribe to SQL (data normalization). To request the same record again, set MORE_RECS_IND to Y.
RECORD_LEN	Provides the length of the record buffer passed. Change RECORD_LEN if the Extract SOURCE_FILE parameter is altered to reflect the new length of the buffer.
RECORD_TYPE	Identifies the record as SQL or ENS for Enscribe.
SOURCE_FILE	Identifies the type of record passed to the exit during Extract processing. When custom mapping is performed, it is sometimes appropriate for the exit routines to change this field. For example, if the exit uses the record buffer from $DATA5.FLS.ACCTFL and changes it to the format of $DATA6.TABS.ACCTTAB, your exit should change SOURCE_FILE to $DATA6.TABS.ACCTTAB.
UPDATE_TYPE	Determines the type of operation associated with the record buffer. Operations types are: • INSERT_IO The record is an insert to the source file. • DELETE_IO The record is a delete, and the buffer is the image of the record being deleted. • UPDATE_IO The record is an update to the source file. The full after update image is supplied. • UPDATE_COMP_ENSCRIBE The record is a compressed Enscribe update. Only primary key fields and changed fields are supplied. Routines are supplied to overlay each field in its corresponding uncompressed location. • UPDATE_COMP_PK_SQL_VAL The record is a primary key update. It contains three parts in the data portion of the record: 1) the before-image key length, 2) the before-image key values in field comp format, and 3) the after-image in field comp format. • UPDATE_COMP_SQL The record is a compressed SQL update. Only primary key fields and changed fields are supplied. Routines are supplied to overlay each field in its corresponding uncompressed location.

EXIT_REC_BUF

EXIT_REC_BUF contains the record about to be processed and is supplied when the call type is PROCESS_RECORD or DISCARD_RECORD. Exit routines can change the contents of this buffer, for example, to perform custom mapping functions. When changing the contents of EXIT_REC_BUF, change the SOURCE_FILE (for Extract only) and RECORD_LEN parameters appropriately so that external processes can identify the record.

Calling environment functions

Environment functions are routines the exit can call to retrieve context information from Extract or Replicat. These functions are summarized below and each of the functions is explained in more detail in the remainder of the chapter. Syntax examples are included for `C`, `TAL`, and `COBOL`.

> **Note:**
>
> Each `COBOL` syntax section documents the `?CONSULT =EXTRACT (or =REPLICAT)` compiler directive. Actually, this directive is declared only once no matter how many functions are called in the user exit. Make sure it points to the correct object type, which will be Extract for an Extract user exit, Replicat for a `TNS` Replicat user exit or `REPR` for native relinkable Replicat user exit.

Function Summary

Function	Description
COMPRESS_RECORD | COMPRESS_RECORD2	Use when some of a target table's columns are present after mapping. Typically, use in conjunction with `DECOMPRESS_RECORD`.
DECOMPRESS_RECORD | DECOMPRESS_RECORD2	Makes compressed SQL updates easier to process and map.
FETCH_CURRENT_RECORD	Obtains the current Enscribe key-sequenced or entry-sequenced target record.
FETCH_CURRENT_RECORD_WITH_LOCK	Obtains the current Enscribe key-sequenced or entry-sequenced target record while locking the record for update.
GET_ALTKEY_INFO	Returns information on a file's alternate key.
GET_COLUMN_INDEX	Given a column name, returns the column index number.
GET_COLUMN_NAME	Given a column index, returns the column's name.
GET_ENV_VALUE	Returns the source or target file name in internal or external format.
GET_EXTRBA	Gets the current position in the extract trail.
GET_EXTSEQNO	Gets the sequence number of the current extract trail.
DATEDIFF	Retrieves the source or target file name.
GET_FOPEN_NUM	Gets the open number of the current file.

Function	Description
GET_NUM_COLUMNS	Returns the number of columns.
GET_RECORD | GET_RECORD2	Use for custom field conversions that may not be handled by Extract or Replicat.
GET_RECORD_LENGTH | GET_RECORD_LENGTH2	Retrieves the length of the target record.
GET_SYSKEY_LENGTH	Retrieves the SYSKEY length for the current record.
GET_TRANSACTION_IND	Supplies a data records position in a transaction, such as first, last, or middle.
GET_USER_TOKEN_VALUE	Allows user to retrieve the value of a user token.
GGS_EMSMESSAGE	Allows a user exit to write an EMS message.
GGS_REPORTMESSAGE	Allows a user exit to write to the report file.
SET_TARGET_RECORD2	Sets the values of a modified record.
SET_TARGET_RECORD_LENGTH2	Sets the length of a modified record.

The function RESULT

The `RESULT` value returned by each function indicates whether or not the function was successful. Possible values are listed below, but different functions may be limited in those they will use.

Return value	Description
`EXIT_ENV_SOURCE_FILE`	Indicates the 24-byte internal format source file name.
`EXIT_ENV_SOURCE_FILE_EXT`	Indicates the source file name in external format.
`EXIT_ENV_TARGET_FILE`	Indicates the target file name in internal format.
`EXIT_ENV_TARGET_FILE_EXT`	Indicates the target file name in external format.
`EXIT_FN_RET_OK`	Function succeeded.
`EXIT_FN_RET_FETCH-ERROR`	An error occurred when attempting to fetch a record.
`EXIT_FN_RET_INVALID_COLUMN`	A non-existent column was referred to in the function call.

Return value	Description
EXIT_FN_RET_INVALID_CONTEXT	Function called at improper time.
EXIT_FN_RET_NOT_SUPPORTED	The request is not supported.
EXIT_FN_RET_INVALID_PARAM	An invalid parameter was passed to the function.

User exit libraries

Three copy libraries define error codes, constants, and structures that are passed to and from the user exits. C user exits should use XLIBC and TAL should use XLIBTAL. For COBOL the library is XLIBCOB.

The function prototypes for C user exits are defined in the library usrdecs. For TAL the library is usrdect. These should be included in the user exit routine.

COMPRESS_RECORD | COMPRESS_RECORD2

Use COMPRESS_RECORD after processing a decompressed record. DECOMPRESS_RECORD is typically invoked when missing column values need to be fetched, such as before mapping a compressed update. COMPRESS_RECORD is called after the mapping or other processing has been completed.

In a C program, precede the INVOKE statement by the #pragma SQL CHAR_AS_ARRAY.

COMPRESS_RECORD is only valid for use with records that are less than 32767. COMPRESS_RECORD2 is valid for both these shorter records and the longer records defined with DDL2.

Syntax COMPRESS_RECORD

For C:

```
#include "usrdecs"
char     *compressed_rec;
short    compressed_len;
char     *decompressed_rec;
short    *columns_present;
short    source_or_target;
short    result;
result = COMPRESS_RECORD (compressed_rec, compressed_len, decompressed_rec,
                          columns_present, source_or_target);
```

For TAL:

```
?source usrdect
int result;
string .ext compressed_rec;
int    .ext compressed_len;
string .ext decompressed_rec;
int    .ext columns_present;
int     source_or_target;
result := COMPRESS_RECORD (compressed_rec, compressed_len, decompressed_rec,
                           columns_present, source_or_target) ;
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 compressed-rec   PIC X(32767).
01 compressed-len   PIC S9(4) COMP.
01 decompressed-rec PIC X(32767).
01 columns-present  PIC S9(4) COMP OCCURS 2000 TIMES.
01 source-or-target PIC S9(4) COMP.
01 result           PIC S9(4) COMP.
ENTER C "COMPRESS_RECORD" using compressed-rec, compressed-len,
         decompressed-rec, columns-present, source-or-target giving result.
```

columns_present
columns-present
An array indicating which columns to compress. For example, if the first, third and sixth columns are to be compressed, and the total number of columns is seven, the columns array should contain `1,0,1,0,0,1,0`. The number of columns in the table can be obtained by `GET_NUM_COLUMNS`. A column that is present but set to `NULL` should be indicated by a `1`.

compressed_len
compress-len
The returned length of the compressed record.

compressed_rec
compressed-rec
The record returned in compressed format. The size is allocated by the user up to the maximum of 32767.

decompressed_rec
decompressed-rec
The record after it has been decompressed. The size is allocated by the user up to the maximum of 32767.

result
A code indicating whether the call was successful or not.

source_or_target
source-or-target
Represented by either `EXIT-FN-SOURCE-VAL` or `EXIT-FN-TARGET-VAL` to indicate whether the record is a source or target record.

Syntax COMPRESS_RECORD2

For C:

```
#include "usrdecs"
char     *compressed_rec;
long     compressed_len;
char     *decompressed_rec;
short    *columns_present;
short    source_or_target;
short    result;
result = COMPRESS_RECORD2 (compressed_rec, compressed_len, decompressed_rec,
                           columns_present, source_or_target);
```

For TAL:

```
?source usrdect
int result;
```

```
string .ext compressed_rec;
int(32).ext compressed_len;
string .ext decompressed_rec;
int    .ext columns_present;
int     source_or_target;
result := COMPRESS_RECORD2 (compressed_rec, compressed_len, decompressed_rec,
                            columns_present, source_or_target) ;
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 compressed-rec   PIC X(128000).
01 compressed-len   PIC S9(8) COMP.
01 decompressed-rec PIC X(128000).
01 columns-present  PIC S9(4) COMP OCCURS 2000 TIMES.
01 source-or-target PIC S9(4) COMP.
01 result           PIC S9(4) COMP.
ENTER C "COMPRESS_RECORD2" using compressed-rec, compressed-len,
         decompressed-rec, columns-present, source-or-target giving result.
```

columns_present
columns-present
An array indicating which columns to compress. For example, if the first, third and sixth columns are to be compressed, and the total number of columns is seven, the columns array should contain `1,0,1,0,0,1,0`. The number of columns in the table can be obtained by `GET_NUM_COLUMNS`. A column that is present but set to `NULL` should be indicated by a `1`.

compressed_len
compress-len
The returned length of the compressed record.

compressed_rec
compressed-rec
The record returned in compressed format. Up to `X(128000)` can be allocated by the user as long as it is declared in extended storage.

decompressed_rec
decompressed-rec
The record after it has been decompressed. Up to `X(128000)` can be allocated by the user as long as it is declared in extended storage.

result
A code indicating whether the call was successful or not.

source_or_target
source-or-target
Represented by either `EXIT-FN-SOURCE-VAL` or `EXIT-FN-TARGET-VAL` to indicate whether the record is a source or target record.

DECOMPRESS_RECORD | DECOMPRESS_RECORD2

`DECOMPRESS_RECORD` makes compressed update records easier to process and map. Typically, `DECOMPRESS_RECORD` is invoked before mapping a compressed update to trigger fetching of missing column values. `COMPRESS_RECORD` is called after processing the compressed updates.

In a C program, precede the INVOKE statement by the #pragma SQL CHAR_AS_ARRAY. Within this structure there may be one or more columns without any true values. Valid columns are indicated in the COLUMNS-PRESENT array.

DECOMPRESS_RECORD is only valid for use with records that are less than 32767. DECOMPRESS_RECORD2 is valid for both these shorter records and the longer records defined with DDL2.

Syntax DECOMPRESS_RECORD

For C:

```
#include "usrdecs"
char     *compressed_rec;
short    compressed_len;
char     *decompressed_rec;
short    *columns_present;
short    source_or_target;
short    result;
result = DECOMPRESS_RECORD (compressed_rec, compressed_len, decompressed_rec,
                            columns_present, source_or_target);
```

For TAL:

```
?source usrdect
int result;
string .ext compressed_rec;
int    .ext compressed_len;
string .ext decompressed_rec;
int    .ext columns_present;
int     source_or_target;
result := DECOMPRESS_RECORD (compressed_rec, compressed_len,
                        decompressed_rec, columns_present, source_or_target);
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 compressed-rec         PIC X(32767).
01 compressed-len         PIC S9(4) COMP.
01 decompressed-rec       PIC X(32767).
01 columns-present.
  05 columns-present-flag PIC S9(4) COMP OCCURS 2000 TIMES.
01 source-or-target       PIC S9(4) COMP.
01 result                 PIC S9(4) COMP.
ENTER C "DECOMPRESS_RECORD" using compressed-rec, compressed-len,
          decompressed-rec, columns-present, source-or-target giving result.
```

columns_present
columns-present
An array of values that indicate the columns present in the compressed record. For example, if the first, third and sixth columns exist in the compressed record, and the total number of columns is seven, the array should contain 1,0,1,0,0,1,0. The number of columns in the table can be obtained by GET_NUM_COLUMNS. A column that is present but set to NULL should be indicated by a 1.

compressed_len
compressed-len
The length of the compressed record.

compressed_rec
compressed-rec
The record in compressed format. The size is allocated by the user up to the maximum of 32767.

decompressed_rec
decompressed-rec
The record returned in decompressed format. The size is allocated by the user up to the maximum of 32767

result
A code indicating whether the call was successful or not.

source_or_target
source-or-target
Represented by either `EXIT-FN-SOURCE-VAL` or `EXIT-FN-TARGET-VAL` to indicate whether the record is a source or target record.

Syntax DECOMPRESS_RECORD2

For C:

```
#include "usrdecs"
char     *compressed_rec;
long     compressed_len;
char     *decompressed_rec;
short    *columns_present;
short    source_or_target;
short    result;
result = DECOMPRESS_RECORD2 (compressed_rec, compressed_len, decompressed_rec,
                             columns_present, source_or_target);
```

For TAL:

```
?source usrdect
int result;
string .ext compressed_rec;
int(32).ext compressed_len;
string .ext decompressed_rec;
int    .ext columns_present;
int     source_or_target;
result := DECOMPRESS_RECORD2 (compressed_rec, compressed_len,
                         decompressed_rec, columns_present, source_or_target);
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 compressed-rec         PIC X(128000).
01 compressed-len         PIC S9(8) COMP.
01 decompressed-rec       PIC X(128000).
01 columns-present.
  05 columns-present-flag PIC S9(4) COMP OCCURS 2000 TIMES.
01 source-or-target       PIC S9(4) COMP.
01 result                 PIC S9(4) COMP.
ENTER C "DECOMPRESS_RECORD2" using compressed-rec, compressed-len,
          decompressed-rec, columns-present, source-or-target giving result.
```

columns_present
columns-present
An array of values that indicate the columns present in the compressed record. For example, if the first, third and sixth columns exist in the compressed record, and the total number of columns is seven, the array should contain `1,0,1,0,0,1,0`. The number of columns in the table can be obtained by `GET_NUM_COLUMNS`. A column that is present but set to `NULL` should be indicated by a `1`.

compressed_len
compressed-len
The length of the compressed record.

compressed_rec
compressed-rec
The record in compressed format. Up to `X(128000)` can be allocated by the user as long as it is declared in extended storage.

decompressed_rec
decompressed-rec
The record returned in decompressed format. Up to `X(128000)` can be allocated by the user as long as it is declared in extended storage.

result
A code indicating whether the call was successful or not.

source_or_target
source-or-target
Represented by either `EXIT-FN-SOURCE-VAL` or `EXIT-FN-TARGET-VAL` to indicate whether the record is a source or target record.

FETCH_CURRENT_RECORD

Use the `FETCH_CURRENT_RECORD` function to get the record from the target table with the same key as the current source record without locking the record. This makes the record buffer available to be read into the user exit by completing a call to `GET_RECORD` for the target image.

`FETCH_CURRENT_RECORD` is supported only for key-sequenced and entry-sequenced Enscribe files. The implementation for entry-sequenced files requires the use of a specified `ALTKEY` for proper positioning.

Fetching from the target table is only supported by Replicat, and only if a mapped target buffer is available. Replicat fetches the current record by key.

Syntax

For C:

```
#include "usrdecs"
short     result;
typedef struct {
  long error_num;
  char error_msg [600];
  long max_length;
  long actual_length;
  short msg_truncated;
} error_info_def;
```

```
error_info_def error_info_ptr;
result = FETCH_CURRENT_RECORD (&error_info_ptr);
```

For TAL:

```
?source usrdect
int result;
int .ext error_info_ptr(error_info_def);
result := FETCH_CURRENT_RECORD(error_info_ptr);
```

For COBOL:

```
?CONSULT =REPLICAT
01 result          PIC S9(4) COMP.
01 error-info.
  02 error-num     PIC S9(4) COMP.
  02 error-msg     PIC X(600).
  02 max-length    PIC S9(4) COMP.
  02 actual-length PIC S9(4) COMP.
  02 msg-truncated PIC S9(4) COMP.
ENTER C "FETCH_CURRENT_RECORD" using error-info giving result.
```

actual-length
The actual length of the error message that is included.

error-msg
The message explaining the error. The size is set when the function is used. `PIC X(600)` and `char [600]` are only examples of a possible value.

error-num
A code indicating whether the call was successful or not.

max-length
The maximum length allowed for an error message.

msg-truncated
Indicates whether the error message has been truncated.

result
A code indicating whether the call was successful or not.

FETCH_CURRENT_RECORD_WITH_LOCK

Use `FETCH_CURRENT_RECORD_WITH_LOCK` to get the record from the target table with the same key as the current source record, locking the record for update. This makes the record buffer available to be read into the user exit by completing a call to `GET_RECORD` for the target image.

`FETCH_CURRENT_RECORD_WITH_LOCK` is supported only for key-sequenced and entry-sequenced Enscribe files. The implementation for entry-sequenced files requires the use of a specified `ALTKEY` for proper positioning. The use of `FETCH_CURRENT_RECORD_WITH_LOCK` is not recommended for inserts into entry-sequenced files.

Fetching from the target table is only supported by Replicat, and only if a mapped target buffer is available. Replicat fetches the current record by key.

Syntax

For C:

```
#include "usrdecs"
short     result;
typedef struct {
  long error_num;
  char error_msg;
  long max_length;
  long actual_length;
  short msg_truncated;
} error_info_def;
error_info_def error_info_ptr;
result = FETCH_CURRENT_RECORD_WITH_LOCK(&error_info_ptr);
```

For TAL:

```
?source usrdect
int result;
int .ext error_info_ptr(error_info_def);
result := FETCH_CURRENT_RECORD_WITH_LOCK(error_info_ptr);
```

For COBOL:

```
?CONSULT =REPLICAT
01 result          PIC S9(4) COMP.
01 error-info.
  02 error-num     PIC S9(4) COMP.
  02 error-msg     PIC X(600).
  02 max-length    PIC S9(4) COMP.
  02 actual-length PIC S9(4) COMP.
  02 msg-truncated PIC S9(4) COMP.
ENTER C "FETCH_CURRENT_RECORD" using error-info giving result.
```

actual-length
The actual length of the error message that is included.

error-msg
The message explaining the error. The size is set when the function is used. `PIC X(600)` is only an example of a possible value.

error_num
A code indicating whether the call was successful or not.

max-length
The maximum length allowed for an error message.

msg-truncated
Indicates whether the error message has been truncated.

GET_ALTKEY_INFO

Returns information about an alternate key.

Syntax

For C:

```
short  result;
short  keytag;
short  keyoff;
short  keylen;
result = GET_ALTKEY_INFO (&keytag, &keyoff, &keylen);
```

For TAL:

```
?source usrdect
int result;
int .ext keytag;
int .ext keyoff;
int .ext keylen;
result := GET_ALTKEY_INFO(keytag, keyoff, keylen);
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 result PIC S9(4) COMP.
01 keytag PIC S9(4) COMP.
01 keyoff PIC S9(4) COMP.
01 keylen PIC S9(4) COMP.
ENTER C "GET_ALTKEY_INFO" using keytag, keyoff, keylen giving result.
```

keytag
The two-byte, generally alphanumeric, code that identifies the alternate key.

keyoff
The offset of the alternate key.

keylen
The length of the alternate key.

result
A code indicating whether the call was successful or not.

GET_COLUMN_INDEX

Given a column name, returns the column index number.

Syntax

For C:

```
#include "usrdecs"
short    result,
short    col_idx;
char     col_name[36];
short    source_or_target;
result = GET_COLUMN_INDEX(&col_idx, col_name, source_or_target);
```

For TAL:

```
?source usrdect
int result;
int    .ext col_idx;
string .ext col_name;
int     source_or_target;
result := GET_COLUMN_INDEX (col_idx, col_name, source_or_target);
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 result            PIC S9(4) COMP.
01 col-idx           PIC S9(4) COMP.
01 col-name          PIC X(36).
01 source-or-target  PIC S9(4) COMP.
ENTER C "GET_COLUMN_INDEX" using col-idx, col-name, source_or_target,
                           giving result.
```

col_idx
col-idx
A sequential number from 0 to (number of columns - 1) that identifies a column.

col_name
col-name
The name of the column from the SQL catalog.

source_or_target
source-or-target
Represented by either `EXIT-FN-SOURCE-VAL` or `EXIT-FN-TARGET-VAL` to indicate whether the record is a source or target record.

result
A code indicating whether the call was successful or not.

GET_COLUMN_NAME

Given a column index, returns the name.

Syntax

For C:

```
#include "usrdecs"
short    result;
short    col_idx;
char     col_name[36];
short    max_name_len,
short    source_or_target;
result = GET_COLUMN_NAME(col_idx, col_name, max_name_len, source_or_target);
```

For TAL:

```
?source usrdect
int result;
int     col_idx;
string .ext column_name;
int     max_name_len;
int     source_or_target;
result := GET_COLUMN_NAME (col_idx, col_name, max_name_len,
                           source_or_target);
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 result       PIC S9(4) COMP.
01 col-idx      PIC S9(4) COMP.
01 col-name     PIC x(36).
01 max-name-len PIC S9(4) COMP.
```

```
ENTER C "GET_COLUMN_NAME" using col-idx, col-name, max-name-len,
                         source-or-target giving result.
```

col_idx
col-idx
A sequential number from 0 to (number of columns - 1) that identifies a column.

col_name
col-name
The name of the column.

max_name_len
The maximum length of the returned column name.

source_or_target
source-or-target
Represented by either `EXIT-FN-SOURCE-VAL` or `EXIT-FN-TARGET-VAL` to indicate whether the record is a source or target record.

result
A code indicating whether the call was successful or not.

GET_ENV_VALUE

Returns the file name of the source or target file in either internal or external format.

For a successful completion, the type of value is indicated in the result code by one following:

Result Code	Meaning
`EXIT_ENV_SOURCE_FILE`	Return value is the source file name in internal format.
`EXIT_ENV_SOURCE_FILE_EXT`	Return value is the source file name in external format.
`EXIT_ENV_TARGET_FILE`	Return value is the target file name in internal format.
`EXIT_ENV_TARGET_FILE_EXT`	Return value is the target file name in external format.

Syntax

For C:

```
#include "usrdecs"
short    result;
short    source_or_target;
char     buf[100];
long     actuallen;
long     maxlen;
short    truncated;
result = GET_ENV_VALUE (source_or_target, &buf, &maxlen, &actuallen,
                        &truncated);
```

For TAL:

```
?source usrdect
int     result;
int     source_or_target;
string  .ext buf;
int(32) maxlen;
int(32) .ext actuallen;
int     .ext truncated;
result :=  GET_ENV_VALUE(source_or_target, buf, maxlen, actuallen,
                         truncated);
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 result           PIC S9(4) COMP.
01 source-or-target PIC S9(4) COMP.
01 buf              PIC X(100).
01 maxlen           PIC S9(8) COMP.
01 actuallen        PIC S9(8) COMP.
01 truncated        PIC S9(4) COMP.
ENTER C "GET_ENV_VALUE" using source-or-target, buf, maxlen, actuallen,
                         truncated giving result.
```

actuallen
The actual length of the returned buffer

buf
The character buffer that will receive data. The space is allocated when the function is used. `PIC X(100)` and `char buf[100]` are only examples of a possible value.

maxlen
The maximum size that was allocated to the buffer.

source_or_target
source-or-target
Represented by either `EXIT-FN-SOURCE-VAL` or `EXIT-FN-TARGET-VAL` to indicate whether the record is a source or target record.

truncated
Indicates whether the returned data has been truncated.

result
A code indicating whether the call was successful or not.

GET_EXITPARAM_VALUE

Use to access the full value of the `EXITPARAM` string. (`FUNCTION_PARAM` of `EXIT_PARAMS` will only provide up to 256 bytes of any `EXITPARAM`.)

Syntax

For C:

```
#include "usrdecs"
char      *exit_parm;
short     len;
short     result;
result = GET_EXITPARAM_VALUE (exit_parm, &len));
```

For TAL:

```
?source usrdect
int result;
string .ext exit_parm;
int .ext5 len;
result := GET_EXITPARAM_VALUE (exit_parm, len);
```

For COBOL:

```
?CONSULT =EXTRACT (or = REPLICAT)
01 EXIT-PARM PIC X(1000).
01 LEN       PIC S9(4) COMP.
01 RESULT    PIC S9(4) COMP.
ENTER C"GET_EXITPARAM_VALUE" using exit_parm, len giving result.
```

exit_parm
exit-parm
The length of the buffer. This is defined by the user.

len
The maximum size that was allocated to the buffer.

result
A code indicating whether the call was successful or not.

GET_EXTRBA

Gets the current position in the Oracle GoldenGate trail.

Syntax

For C:

```
#include "usrdecs"
long long extrba;
short     result;
result = GET_EXTRBA(&extrba);
```

For TAL:

```
?source usrdect
int     result;
int(64) .ext extrba;
result := GET_EXTRBA(extrba);
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 result S9(4) COMP.
01 extrba S9(16) COMP.
ENTER C "GET_EXTRBA" using &extrba giving result.
```

extrba
The current position in the Oracle GoldenGate trail.

result
A code indicating whether the call was successful or not.

GET_EXTSEQNO

Gets the sequence number for the current Oracle GoldenGate trail.

Syntax

For C:

```
#include "usrdecs"
long     seqno;
short    result;
result = GET_EXTSEQNO(&extseqno);
```

For TAL:

```
?source usrdect
int     result;
int(32) .ext seqno;
result := GET_EXTSEQNO(seqno) ;
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 result     PIC S9(4) COMP.
01 seqno      PIC S9(4) COMP.
ENTER C "GET_EXTSEQNO" using seqno giving result.
```

seqno
The sequence number of the Oracle GoldenGate trail file.

result
A code indicating whether the call was successful or not.

GET_FILENAME

Retrieves the source or target file name.

Syntax

For C:

```
#include "usrdecs"
short result;
typedef struct fname_def
{char volume[8];
  char subvol[9];
  char file[8];
} fname_def;
fname_def fname;
short source_or_target;
result = GET_FILENAME(fname, source_or_target);
```

For TAL:

```
?source usrdect
int result;
int .ext fname;
```

```
int source_or_target;
result :=  GET_FILENAME (fname,  source_or_target);
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 result     PIC S9(4) COMP2.
01 fname.
  02 volume   pic x(8).
  02 subvol   pic x(8).
  02 filename pic x(8).
01 source-or-target PIC S9(4) COMP.
ENTER C "GET_FILENAME" using fname, source-or-target giving result.
```

filename
The name of the current record's file.

fname
The name of the file in internal format.

source_or_target
Represented by either `EXIT-FN-SOURCE-VAL` or `EXIT-FN-TARGET-VAL` to indicate whether the record is a source or target record.

result
A code indicating whether the call was successful or not.

GET_FOPEN_NUM

Returns the open number for the file.

Syntax

For C:

```
#include "usrdecs"
short    result;
short    fopen_num;
result = GET_FOPEN_NUM(&fopen_num);
```

For TAL:

```
?source usrdect
int  result;
int  .ext fopen_num;
result :=  GET_FOPEN_NUM (fopen_num);
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 result    PIC S9(4) COMP.
01 fopen-num PIC S9(4) COMP.
ENTER C "GET_FOPEN_NUM" using fopen-num giving result.
```

fopen_num
fopen-num
The number assigned to the file by the process opening it.

source_or_target
Indicates whether to retrieve the source or target file name.

GET_NUM_COLUMNS

GET_NUM_COLUMNS returns the number of columns in an SQL table. Use this information when processing compressed and decompressed records.

Syntax

For C:

```
#include "usrdecs"
short    num_columns;
short    source_or_target;
short    result;
result = GET_NUM_COLUMNS (&num_columns, source_or_target);
```

For TAL:

```
?source usrdect
int     result;
int     .ext num_columns;
int     source_or_target;
result := GET_NUM_COLUMNS (num_columns, source_or_target);
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 num-columns      PIC S9(4) COMP.
01 source-or-target PIC S9(4) COMP.
01 result           PIC S9(4) COMP.
ENTER C "GET_NUM_COLUMNS" using num-columns, source-or-target
         giving result.
```

num_columns
num-columns
The number of columns in the SQL table.

source_or_target
source-or-target
Indicates whether to retrieve number of columns for the source or target table.

result
A code indicating whether the call was successful or not.

GET_RECORD | GET_RECORD2

Data mapping from a source to a different target format is often required when extracting or replicating data. Although the COLMAP specification supported by both Replicat and Extract works sufficiently for most, user exits may be necessary for some field conversions.

Extract and Replicat pass data records to the exit after converting to the target format. In the event that a few fields do not convert, GET_RECORD and GET_RECORD2 provide a way to obtain information for custom field conversions. For example, an exit could convert a proprietary date field (such as YYDDD) in an Enscribe database to a standard SQL date in the target record, while other columns would be mapped by Extract.

GET_RECORD is only valid for use with record less than 32767. GET_RECORD2 is valid for both these smaller records and the longer records defined with DDL2.

Formats

Deletes, inserts and updates appear in the buffer as full record images. For SQL tables, this is the same format as produced by an INVOKE of the table.

Compressed SQL updates

Compressed SQL updates have the format:

index length value index length value…

Where:

- *index* is a two byte index into the list of columns of the table (first column is zero).
- *length* is the two byte length of the table.
- *value* is the actual column value, including when applicable a two byte null indicator (0 for not null, -1 for null).

Compressed Enscribe updates

Compressed Enscribe updates have the 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 and value is the actual data. Fragments can span field boundaries, so full fields are not always retrieved (unless compression is off or FETCHCOMPS is elected).

Compressed Primary Key Updates

Compressed primary key updates contain three parts:

- The before-image key length
- The before-image key value in field comp format
- The after-image in field comp format

You can modify the record keeping the format intact, or use DECOMPRESS_RECORD and COMPRESS_RECORD to obtain a full record format. To use the callback functions, you will need to separate the before and after-images and use the callback on only one portion at a time.

Use the following to move the record pointer so that you can manipulate the before-image up to the before_len. Note that the size of key length fields must be a long.

```
long before_len;
long after_len;
char *rec_ptr;
memcpy (&before_len, rec, sizeof(before_len));  /* get before len */
rec_ptr = rec + sizeof (before_len);   /* move rec_ptr passed len */
```

For the after-image, use the following to manipulate the after-image up to the after_len.:

```
rec_ptr += before_len;
after_len = exit_params->record_len - sizeof (before_len) - before_len;
```

To see an example of the trail record with its before and after-images, use Logdump with the `Detail` option set on.

Syntax GET_RECORD

For C:

```
#include "usrdecs"
char     *buf;
short    len;
short    io_type;
short    source_or_target;
short    result;
result = GET_RECORD (buf, &len, &io_type, source_or_target);
```

For TAL:

```
?source usrdect
int result;
string .ext buf;
int    .ext len;
int    .ext io_type;
int    source_or_target;
result := GET_RECORD (buf, len, io_type, source_or_target);
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 buf              PIC X(32767).
01 len              PIC S9(4) COMP.
01 io-type          PIC S9(4) COMP.
01 source-or-target PIC S9(4) COMP.
01 result           PIC S9(4) COMP.
ENTER C "GET_RECORD" using buf, len, io-type, source-or-target
                     giving result.
```

buf
The record buffer for data returned by the Extract or Replicat programs. The memory for this buffer must be allocated by the user. Up to `X(32767)` can be used for the buffer.

io_type
io-type
Indicates the type of operation represented by the record, such as:
`3` — Delete
`5` — Insert
`10` — Update
`11` — Compressed Enscribe Update
`15` — Compressed SQL Update
`115` — Compressed primary key Update

len
`LEN` is the length of the data returned in `BUF`.

source_or_target
source-or-target
Represented by either EXIT-FN-SOURCE-VAL or EXIT-FN-TARGET-VAL to indicate whether to retrieve the source or target record.

result
A code indicating whether the call was successful or not.

Syntax GET_RECORD2

For C:

```
#include "usrdecs"
char     *buf;
long    len;
short    io_type;
short    source_or_target;
short    result;
result = GET_RECORD2 (buf, &len, &io_type, source_or_target);
```

For TAL:

```
?source usrdect
int result;
string .ext buf;
int(32).ext len;
int    .ext io_type;
int    source_or_target;
result := GET_RECORD2 (buf, len, io_type, source_or_target);
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 buf              PIC X(128000).
01 len              PIC S9(8) COMP.
01 io-type          PIC S9(4) COMP.
01 source-or-target PIC S9(4) COMP.
01 result           PIC S9(4) COMP.
ENTER C "GET_RECORD2" using buf, len, io-type, source-or-target
                     giving result.
```

buf
The record buffer for data returned by the Extract or Replicat programs. The memory for this buffer must be allocated by the user. Up to X(128000) can be used in the buffer as long as it is declared in extended storage.

io_type
io-type
Indicates the type of operation represented by the record, such as:
3 — Delete
5 — Insert
10 — Update
11 — Compressed Enscribe Update
15 — Compressed SQL Update
115 — Compressed primary key Update

len
LEN is the length of the data returned in BUF.

source_or_target
source-or-target
Represented by either EXIT-FN-SOURCE-VAL or EXIT-FN-TARGET-VAL to indicate whether to retrieve the source or target record.

result
A code indicating whether the call was successful or not.

GET_RECORD_LENGTH | GET_RECORD_LENGTH2

When returning control to Extract or Replicat, user exits are responsible for setting the length of the target record if and when mapping occurs. Use GET_RECORD_LENGTH or GET_RECORD_LENGTH2 to obtain this value when the mapped record is not compressed. Set the EXIT-PARAMS structure member RECORD-LENGTH to this value.

GET_RECORD_LENGTH2 is only valid for use with records less than 32767. GET_RECORD_LENGTH2 is valid for both these smaller records and the longer records defined with DDL2.

Syntax GET_RECORD_LENGTH

For C:

```
#include "usrdecs"
short    record_length;
short    source_or_target;
short    result;
result = GET_RECORD_LENGTH (&record_length, source_or_target);
```

For TAL:

```
?source usrdect
int     result;
int     .ext record_length;
int     source_or_target;
result := GET_RECORD_LENGTH(record_length,  source_or_target);
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 record-length    PIC S9(4) COMP.
01 source-or-target PIC S9(4) COMP.
01 result           PIC S9(4) COMP.
ENTER C "GET_RECORD_LENGTH" using record-length, source-or-target
         giving result.
```

record_length
record-length
The record length as calculated internally by Extract or Replicat.

source_or_target
source-or-target
Represented by either EXIT-FN-SOURCE-VAL or EXIT-FN-TARGET-VAL to indicate whether to retrieve the source or target record.

result
A code indicating whether the call was successful or not.

Syntax GET_RECORD_LENGTH2

For C:

```
#include "usrdecs"
long    record_length;
short    source_or_target;
short    result;
result = GET_RECORD_LENGTH2 (&record_length, source_or_target);
```

For TAL:

```
?source usrdect
int     result;
int(32) .ext record_length;
int     source_or_target;
result := GET_RECORD_LENGTH2(record_length,  source_or_target);
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 record-length    PIC S9(8) COMP.
01 source-or-target PIC S9(4) COMP.
01 result           PIC S9(4) COMP.
ENTER C "GET_RECORD_LENGTH2" using record-length, source-or-target
         giving result.
```

record_length
record-length
The record length as calculated internally by Extract or Replicat.

source_or_target
source-or-target
Represented by either `EXIT-FN-SOURCE-VAL` or `EXIT-FN-TARGET-VAL` to indicate whether to retrieve the source or target record.

result
A code indicating whether the call was successful or not.

GET_SYSKEY_LENGTH

Returns the length of `SYSKEY` for the current record.

Syntax

For C:

```
include "usrdecs"
short   syskeylen,
short   source_or_target;
short   result;
result = GET_SYSKEY_LENGTH(&syskey_len, source_or_target);
```

For TAL:

```
?source usrdect
int  result;
int  .ext syskeylen;
```

```
int  source_or_target;
result := GET_SYSKEY_LENGTH (syskeylen, source_or_target) ;
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 result           PIC S9(4) COMP.
01 syskeylen        PIC S9(4) COMP.
01 source_or_target PIC S9(4) COMP.
ENTER C "GET_SYSKEY_LENGTH" using syskeylen, source-or-target
         giving result.
```

syskeylen
The length of the system key.

source_or_target
source-or-target
Represented by either `EXIT-FN-SOURCE-VAL` or `EXIT-FN-TARGET-VAL` to indicate whether to retrieve the source or target record.

result
A code indicating whether the call was successful or not.

GET_TRANSACTION_IND

Knowing whether a data record is the first, last or middle operation in a transaction can prove useful to an exit routine. For example, an exit might want to compile the details of each transaction and output a special summary record at the end. This type of processing is accommodated by `GET_TRANSACTION_IND`.

Syntax

For C:

```
#include "usrdecs"
short result;
short trans_ind;
result = GET_TRANSACTION_IND (&trans_ind);
```

For TAL:

```
?source usrdect
int result;
int .ext trans_ind;
result := GET_TRANSACTION_IND (trans_ind);
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 result    PIC S9(4) COMP.
01 trans-ind PIC S9(4) COMP.
ENTER C "GET_SOURCE_RECORD" using trans-ind giving result.
```

trans_ind
trans-ind
Indicates whether the current record is the first (`0`), middle (`1`), last (`2`), or both first and last (`3`) record in a transaction.

result
A code indicating whether the call was successful or not.

GET_USER_TOKEN_VALUE

GET_USER_TOKEN_VALUE allows the user exit to retrieve the value of user tokens.

Buffers

xlibc

```
typedef struct __user_token_value {
   char   token_name[256];
   short  actual_length;
   char   token_value_buf[2000];
} user_token_vaue_def;
```

xlibcob

```
01 result native-2.
?Section TOKENS,Tandem
  01 USER-TOKEN-VALUE.
    02 TOKEN-NAME PIC X(256).
    02 ACTUAL-LENGTH NATIVE-2.
    02 TOKEN-VALUE-BUF PIC X(2000).
```

Syntax

For C:

```
#include "usrdecs"
#include "xlibc"
short result = 0;
user_token_value_def usr_token_val;
result = GET_USER_TOKEN_VALUE (&usr_token_val);
```

For TAL:

```
?source usrdect
int result;
int .ext utoken_value_ptr(user_token_value_def);
result := GET_USER_TOKEN_VALUE(utoken_value_ptr);
```

For COBOL:

```
?ENV COMMON
?CONSULT =EXTRACT (or =REPLICAT)

ENTER C "GET_USER_TOKEN_VALUE" using USER-TOKEN-VALUE giving RESULT.
```

Return Values

```
EXIT_FN_RET_OK
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_TOKEN_NOT_FOUND
```

token_name
TOKEN-NAME
Up to 256 characters to specify the name of the user token.

actual_length
The actual length of the data returned in the buffer.

token_value_buf
The 2000 byte buffer that will receive data.

result
A code indicating whether the call was successful or not. One of the following:

```
EXIT_FN_RET_OK
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_TOKEN_NOT_FOUND
```

GGS_EMSMESSAGE

GGS_EMSMESSAGE allows a user exit to write a message to the EMS log file. The message will have the Oracle GoldenGate subsystem ID.

Syntax

For C:

```
#include "usrdecs"
short result;
int   evtnum;
int   severity;
char  *text;
result = GGS_EMSMESSAGE(int evtnum, text, severity);
```

For TAL:

```
?source usrdect
int    result;
int    evtnum;
string .ext text;
int    severity;
result := GGS_EMSMESSAGE(evtnum, text, severity);
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 evtnum        PIC S(4) COMP.
01 result        PIC S(4) COMP.
01 text          PIC X(132).
01 severity      PIC S(4) COMP.
ENTER C "GGS_EMSMESSAGE" using evtnum, text, severity giving result.
```

evtnum
An arbitrary number used to identify a message.

text
The text of the message.

severity
A value that identifies how severe the message is, as in:

- MSG_INFO 1 — normal EMS message.

- `MSG_CRITICAL 2` — EMS message with emphasis set to `ON`.
- `MSG_FATAL 3` — abends the process after writing the message.

result
A code indicating whether the call was successful or not.

GGS_REPORTMESSAGE

Allows a user exit to write a message to the report file.

Syntax

For C:

```
#include "usrdecs"
short    result;
char    text[132];
result = GGS_REPORTMESSAGE(&text);
```

For TAL:

```
?source usrdect
int result;
string .ext text;
result := GGS_REPORTMESSAGE(text) ;
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 result PIC S9(4) COMP.
01 text   PIC X(132).
ENTER C "GGS_REPORTMESSAGE" using text giving result.
```

> **Note:**
>
> The string value in message text must be null terminated.

text
The text of the message.

result
A code indicating whether the call was successful or not.

SET_TARGET_RECORD2

Use `SET_TARGET_RECORD2` to set the values of a modified record. This step is not required for the buffers under 32 K, but is required for larger records defined with DDL2.

Syntax

For C:

```
#include "usrdecs"
char     *buf;
long     record_len;
```

```
short    io_type;
short    result;
result = SET_TARGET_RECORD2 (buf, &record_len, &io_type);
```

For TAL:

```
?source usrdect
string  .ext buf;
int(32) len;
int     io_type;
int     result;
result := SET_TARGET_RECORD2 (buf, len, io_type);
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 buf       PIC X(128000).
01 len       PIC S9(8) COMP.
01 io-type   PIC S9(4) COMP.
01 result    PIC S9(4) COMP.
ENTER C "SET_TARGET_RECORD2" using buf, len, io-type giving result.
```

buf
The record buffer for data returned by the Extract or Replicat programs. The memory for this buffer must be allocated by the user. Up to `X(127000)` can be used in the buffer as long as it is declared in extended storage.

len
`LEN` is the length of the data returned in `BUF`.

io_type
io-type
Indicates the type of operation represented by the record, such as:
`3` — Delete
`5` — Insert
`10` — Update
`11` — Compressed Enscribe Update
`15` — Compressed SQL Update
`115` — Compressed primary key Update

result
A code indicating whether the call was successful or not.

SET_TARGET_RECORD_LENGTH2

Use `SET_TARGET_RECORD_LENGTH2` to set the length of a modified target record. This step is not required for the smaller records defined with DDL, but is required for large records defined with DDL2.

Syntax

For C:

```
#include "usrdecs"
long     record_len;
short    result;
result = SET_TARGET_RECORD_LENGTH2 (&record_len);
```

For TAL:

```
?source usrdect
int(32) record_len;
int     io_type;
int     result;
result := SET_TARGET_RECORD_LENGTH2 (record_len);
```

For COBOL:

```
?CONSULT =EXTRACT (or =REPLICAT)
01 record-len  PIC S9(8) COMP.
01 result      PIC S9(4) COMP.
ENTER C "SET_TARGET_RECORD_LENGTH2" using record-length giving result.
```

record_len
record-len
The record length as calculated internally by Extract or Replicat.

result
A code indicating whether the call was successful or not.

8
Event Error and Warning Messages

This chapter lists the cause and recommended action for error and warning messages generated by Oracle GoldenGate for HP NonStop on the Guardian platform.

GGS-EVT-STARTUP-PARAM—101
Cause: Issued when an unrecognized, invalid, or missing parameter is encountered during program startup.

Action: Correct the parameter file (the process `IN` file) and restart.

GGS-EVT-AUDIT-MISSING—102
Cause: A TMF audit trail required by Extract is unavailable. Possibly the necessary audit was purged by TMF. If the audit file exists on tape, this can occur when specifying `NORESTORE`. Retrieval of data within the specified time period may not be possible.

Action: Examine the error message for more details. If `NORESTORE` was specified in the parameter file, and the missing file is on tape, consider eliminating this parameter. If the audit cannot be located either on disk or tape, other means of data extraction will be required for the given time period.
Tokens:

`GGS-TKN-AUDIT-FILE`

GGS-EVT-FILE—103
Cause: A Guardian file error was encountered during an operation on the file or process.

Action: Examine the Guardian error number to determine the exact nature of the error. Attempt to correct the error. In some instances, errors are automatically retried if corrected within a short period of time.
Tokens:

`GGS-TKN-FILE`
`GGS-TKN-GUARDIAN-ERR`

GGS-EVT-AUDIT-READ—104
Cause: Issued when an error is encountered while reading a TMF audit trail. This error is generally returned under unusual circumstances.

Action: Examine the audit error code. A `-900`, may indicate a problem restoring the tape (the restore may not have completed). If a `FILEINFO` on the restored audit trail indicates `?`, refer to SNOOP documentation to turn the corrupt flag off. Restart if necessary. If the problem persists, contact Oracle GoldenGate Support.
Tokens:

`GGS-TKN-AUDIT-FILE`
`GGS-TKN-AR-ERR`

GGS-EVT-MEM-ALLOC—105
Cause: Issued due to an internal problem allocating memory.

Action: Restart the program. If the problem persists, contact Oracle GoldenGate.

GGS-EVT-SPI-PROCESS—107
Cause: An internal problem with an SPI procedure call or a problem starting a process. This error can also occur when the Audserv program is improperly secured. To secure Audserv, issue:

```
> LOGON SUPER.SUPER
- FUP
- GIVE AUDSERV, SUPER.SUPER
- SECURE AUDSERV, "NUNU", PROGID
- LICENSE (AUDSERV, TMFARUL2)
- EXIT
```

Action: Note error message and contact Oracle GoldenGate. If the problem is starting a TMFSERVE process, it may be necessary to `FUP LICENSE TMFSERVE`.
Tokens:

```
GGS-TKN-FILE (optional)
GGS-TKN-GUARDIAN-ERR (optional)
```

GGS-EVT-SQL—109
Cause: An error was encountered during an SQL operation.

Action: Dependent on the type of error. If the error occurred during a replicated operation, error detail can be found in the Replicat discard file.

GGS-EVT-ABEND-RECOVERY—110
Cause: After restarting after a process `ABEND`, Extract encountered out-of-sync data since the last successful checkpoint. A `TABLE` or `FILE` entry has likely been added in the Extract parameter file since the last run, or a table catalog definition has changed.

Action: After an `ABEND`, the parameter file must be exactly the same as during the prior run. Restore the parameter file to its prior state. Otherwise, resetting checkpoints (using GGSCI) may be required (review the situation carefully before doing so).

GGS-EVT-EXTRACT-FILE—112
Cause: An error was encountered while processing a file in the Oracle GoldenGate trail. View the message in the event file for more information.

Action: Recovery depends on the error message. If this is a file error (such as a security violation), correct the problem and restart the process.

GGS-EVT-GROUP-DU—113
Cause: Another process is already running with the checkpoint group indicated in the parameter file. This problem is often caused by mistakenly running the same job twice simultaneously.

Action: If the name of the Extract or Replicat parameter is incorrect, fix it. Otherwise, no recovery required.

GGS-EVT-MAXFILES—115
Cause: The maximum files for an was reached during Extract processing.

Action: Dependent on the situation. Possibilities include renaming or purging the oldest files, or increasing the `MAXFILES` option of the parameter using the GGSCI program. After taking one of these actions, restart the Extract program.
Tokens:

`GGS-TKN-FILE`

GGS-EVT-AUDIT-SERVER—116
Cause: Extract could not communicate with an Audserv process. The Audserv process may have been mistakenly stopped.

Action: Restart the Extract program. If the problem persists, contact Oracle Support.

GGS-EVT-USER-EXIT—117
Cause: A logic error probably exists when a user exit routine converts data to another format.

Action: Correct the application logic and rebind the user exit.

GGS-EVT-RMT-FILE—118
Cause: A file error was encountered on a remote system while writing Extract data to disk.

Action: Correct the problem and restart the program.
Tokens:

`GGS-TKN-RMT-FILE`

GGS-EVT-REP-NEW-COLUMN-ERR—119
Cause: Replicat encountered an SQL error delivering an `ADD COLUMN` statement on a target table.

Action: Correct the problem and restart Replicat.
Tokens:

`GGS-TKN-FILE`
`GGS-TKN-SQL-ERR`
`GGS-TKN-GUARDIAN-ERR`

GGS-EVT-WRONG-VERSION—120
Cause: The version of Oracle GoldenGate is incompatible with the current operating system.

Action: Install the correct version of Oracle GoldenGate.

GGS-EVT-MGR-CHILD-EXISTS—122
Cause: The Manager child process was not stopped.

Action: This is usually a warning message, but if this is a irrecoverable error, examine the error message to determine the exact nature of the error (such a security violation). Attempt to correct the error.

GGS-EVT-TCPIP—150
Cause: A TCP/IP error occurred while attempting to write Extract data to a remote system.

Action: If the error is "Connection Refused," start an Extract Collector process on the remote system. Also, check the `RMTHOST` and `PORT` parameters to make sure they match

the Collector startup parameters. TCP/IP errors are frequently retried automatically for a period of time. For more information see TCPIPSWTICHERRS on "TCPIPSWITCHERRS".

Tokens:

GGS-TKN-RMT-HOST
GGS-TKN-TCP-PORT
GGS-TKN-TCP-ERR

GGS-EVT-SYNC-DUPPROC-ABEND—160
Cause: Syncfile abended.

Action: Examine the error message to determine the exact nature of the error. Attempt to correct the error.

GGS-EVT-EXT-ABEND—190
Cause: The Extract process is ending abnormally.

Action: Examine the previous error messages and the Extract report file to determine the exact cause.

GGS-EVT-REP-ABEND—191
Cause: The Replicat process is ending abnormally.

Action: Examine the previous error messages and the Replicat report file to determine the exact cause.

GGS-EVT-MGR-ABEND—192
Cause: The Manager process is ending abnormally.

Action: Examine the previous error messages in the log in order to determine the exact cause.

GGS-EVT-SERVER-ABEND—193
Cause: Collector abended.

Action: Examine the error message to determine the exact nature of the error. Attempt to correct the error.

GGS-EVT-SYNC-ABEND—194
Cause: Syncfile abended.

Action: Examine the error message to determine the exact nature of the error. Attempt to correct the error.

GGS-EVT-TMF-ERROR—195
Cause: Error occurred during the ENDTRANSACTION command to TMF.

Action: Examine the error message to determine the exact nature of the error. Attempt to correct the error. Contact Oracle GoldenGate Support if necessary.

GGS-EVT-FILE-ALTER—200
Cause: The program encountered a FILE ALTER record in the audit trail affecting one of the files designated for extraction. The exact nature of what changed about the file is impossible to determine from this message.

Action: If an extracted file or table definition was changed before all related audit was processed, you may need to re-synchronize data. If audit was turned off in the table or file, it is possible that updates to the table are not extracted. This message can also occur when changes that are not critical to extract are encountered.
Tokens:

`GGS-TKN-FILE`

GGS-EVT-AUDIT-COMPRESSED—204
Cause: The record encountered in the audit trail is compressed and missing a required column. This column may have been used in a `MAP WHERE` clause. The record is discarded.

Action: Examine the compressed image in the discard file. The recovery is application-dependent.
Tokens:

`GGS-TKN-FILE`

GGS-EVT-NO-CONTEXT-REC—207
Cause: The checkpoint database is out-of-sync.

Action: Contact Oracle GoldenGate.

GGS-EVT-RECORD-VERSION—208
Cause: The record encountered in the audit trails is out of synchronization with the current version of the SQL table. Changed data is not output because the versions are different.

Action: Manual re-synchronization (such as reload) may be required.
Tokens:

`GGS-TKN-FILE`

GGS-EVT-DUP-ERROR—209
Cause: The Manager process could not `FUP DUP` an audit trail to a backup volume.

Action: Free up disk space or specify a different `ALTLOC` option in the GGSCI `ADD ATCONFIG` command.

GGS-EVT-FILE-NO-AUDIT—210
Cause: The file or table is not audited. For Extract, this means that data could be missing (if capturing changes from the audit trail). For Replicat, this means that replicated operations to the file or table will occur without TMF protection.

Action: Use FUP or SQLCI to turn audit on the file or table. Data may be missing, if so, other means of data extract and replication may be required. There can also be legitimate reasons for temporarily turning audit off.
Tokens:

`GGS-TKN-FILE`

GGS-EVT-TAPE-RESTORE-NOW—211
Cause: An audit dump needs to be restored from tape in order for Extract to continue processing.

Action: Follow the instructions (displayed by SNOOP) on the operator console for restoring the tape.

Tokens:

GGS-TKN-AUDIT-FILE

GGS-EVT-TAPE-RESTORE—212
Cause: At some point, the specified audit file will need to be restored from tape. This message is issued at startup so that tapes can be restored before needed, enabling faster and more reliable processing.

Action: None.
Tokens:

GGS-TKN-AUDIT-FILE

GGS-EVT-FETCH-RECORD—213
Cause: Extract encountered a compressed update record, attempted to retrieve the entire record from the file or table and it no longer exists.

Action: This is generally a tolerable condition that occurs when an update to a database record is followed by a delete.

GGS-EVT-TRANS-ABORTED—220
Cause: An error occurred while Replicat was executing REPERROR, TRANSABORT.

Action: Examine the error message to determine the exact nature of the error. Attempt to correct the error.

GGS-EVT-MISSING-TRANS-BEGIN—221
Cause: Replicat cannot process the current transaction in the Extract file because the beginning of the transaction is missing. Processing resumes at the next transaction. Possibly, a previous Extract file was purged.

Action: Examine the discard file for the rejected data and use manual procedures to recover the data.

GGS-EVT-MAPPING-PROBLEM—222
Cause: A problem was encountered while mapping a source record to a target format. Usually this means that the source record has unexpected or incorrect data in it. The record is discarded. Depending on how error handling is configured in the parameter file, the process continues, aborts the transaction, or abends. See the Replicat REPERROR parameter.

Action: Examine the discard record to determine what may have caused the mapping problem. Use manual procedures to extract or replicate the necessary data, or alter the mapping specification and restart.
Tokens:

GGS-TKN-SRC-FILE
GGS-TKN-DST-FILE

GGS-EVT-ROLLOVER—224
Cause: Extract closed a file in an sequence and opened the next one.

Action: If this message is highlighted, take action before the next rollover to ensure that Extract processing continues smoothly. The oldest file in the sequence will need to be renamed or purged (use caution).
Tokens:

GGS-TKN-FILE

GGS-EVT-PURGE-EXTRACT—226
Cause: The Manager process purged an Extract file that is no longer required by any Replicat processes.

Action: None required. Informational.
Tokens:

GGS-TKN-FILE

GGS-EVT-PURGE-LOG—229
Cause: The Manager process recycled the oldest Logger trail file to make room for new log records.

Action: None required. Informational.
Tokens:

GGS-TKN-LOG-FILE

GGS-EVT-CREATE-LOG-FAILED—230
Cause: GGSCI or Manager failed to create a Logger trail file. This error can be caused by not enough disk space on a volume or de-allocated space in the trail files.

Action: This error is critical. To avoid it, do not deallocate unused extents in the Logger trail files. Log file space is pre-allocated to ensure availability when needed. Also, try purging other files in the system to make room for the newest log files. Reconfigure `LOG` space to use smaller extent sizes.
Tokens:

GGS-TKN-LOG-FILE
GGS-TKN-GUARDIAN-ERR

GGS-EVT-MGR-STOPPED—231
Cause: The Manager process stopped gracefully, as requested using GGSCI. Manager operations are critical to logging and other functions, so should be restarted within a reasonable time frame, assuming other Oracle GoldenGate operations are continuing.

Action: None required. Informational.

GGS-EVT-NO-UPDATE-COLUMNS—232
Cause: No columns are specified for update in the `MAP` statement. Typically the data source is not a NonStop source, and has bad records

Action: Check the remote Extract parameter file and `TRANDATA` for primary key logging.

GGS-EVT-ABNORMAL-STOP—233
Cause: The user entered a `STOP` command from GGSCI using `FORCESTOP`.

Action: Information only.

GGS-EVT-SYNC-DUP-FILESET-ERR—252
Cause: The Syncfile program statistics are reporting errors. The message may be similar to:

```
GGS WARNING 252 Syncfile errors/warnings duplicating $data02.tssparm.*
to $data01.tssparm.* (total=34, excluded=0, modified=34, attempted=34,
failed=15).
```

Action: Warning only.

GGS-EVT-SYNC-DUP-FILE-ERR—253
Cause: Syncfile failed to duplicate the file.

Action: Warning. Make sure the file is available or named correctly. Re-run Syncfile if necessary.

GGS-EVT-COLL-OPEN-FILE-ERR—261
Cause: A file error was encountered during an open of a file by the Collector.

Action: Examine the error number to determine the exact nature of the error (for example, a security violation or disk is full). Attempt to correct the error and restart Extract on the source system.

GGS-EVT-COLL-IO-ERR—262
Cause: Collector attempted to position in a disc file after `EOF`.

Action: Extract will try to re-send the data. If the problem continues Extract abends and re-starts. If Extract exceeds the maximum restarts and the problem remains, delete the Extract and Replicat processes and re-add them.

GGS-EVT-REP-WAIT-CANCELLED—280
Cause: The user requested, through GGSCI, to cancel waiting on a file event in Replicat (negating the `WAITFILEEVENT` parameter).

Action: None. Used to trace processing activity.

GGS-EVT-REP-WAIT-CRITICAL—281
Cause: Replicat has been waiting a significant amount of time for a file event (such as a `RENAME` or `DUP`) to occur.

Action: Waiting for a file event holds up Replicat processing. You can cancel the wait using the `SEND REPLICAT` *`group_name`*, `BYPASSFILEEVENT` GGSCI command. Deciding to cancel a wait is an application-specific issue and should be considered carefully.

GGS-EVT-PROCESSED-MARKER—300
Cause: An audit marker record created by GGSCI was processed by Extract or Replicat.

Action: None. Markers are used to mark application-specific points of interest within the audit trails. See discussions of markers elsewhere in this manual for more information.

GGS-EVT-USER-CMD—301
Cause: A command was received by the Extract or Replicat process.

Action: None. Used to trace processing activity.

GGS-EVT-REP-STARTED—302
Cause: The Replicat process started.

Action: None. Used to trace processing activity.

GGS-EVT-EXT-STARTED—303
Cause: The Extract process started.

Action: None. Used to trace processing activity.

GGS-EVT-FILE-ERR-RECOVERED—304
Cause: A file operation that previously encountered an error was retried successfully.

Action: None. Indicates that processing has resumed normally after problem correction.
Tokens:

`GGS-TKN-FILE`

GGS-EVT-TCP-ERR-RECOVERED—305
Cause: A TCP/IP operation that previously encountered an error was retried successfully.

Action: None. Indicates that processing has resumed normally after problem correction.
Tokens:

`GGS-TKN-RMT-HOST`
`GGS-TKN-TCP-PORT`

GGS-EVT-EXT-STOP-NORMAL—306
Cause: Extract processing terminated normally.

Action: None. Used to trace processing activity.

GGS-EVT-EXT-STOP-USER—307
Cause: Extract processing was terminated from GGSCI by a user.

Action: None. Used to trace processing activity.

GGS-EVT-REP-STOP-NORMAL—308
Cause: Replicat processing terminated normally.

Action: None. Used to trace processing activity.

GGS-EVT-REP-STOP-USER—309
Cause: Replicat processing was terminated from GGSCI by a user.

Action: None. Used to trace processing activity.

GGS-EVT-MGR-PROCESS-RUNNING—310
Cause: The Manager process is reporting that a process is alive.

Action: None. Used to trace processing activity.

GGS-EVT-MGR-PROCESS-DOWN-OK—311
Cause: The Manager process is reporting that a process terminated normally and is down. The message is marked as critical when the `DOWNCRITICAL` Manager parameter is specified.

Action: If this message is highlighted, restarting the process may be required. Otherwise no recovery action necessary.

GGS-EVT-MGR-PROCESS-ABENDED—312
Cause: The Manager process is reporting that a process abended.

Action: Investigate the reason for the abend, fix the problem and restart the program.

GGS-EVT-MGR-STARTED—313
Cause: The Manager process started.

Action: None. Used to trace processing activity.

GGS-EVT-MGR-MADE-AUDIT-COPY—314
Cause: The Manager process made a backup copy of an audit trail file.

Action: None. Informational.
Tokens:

`GGS-TKN-AUDIT-FILE`

GGS-EVT-MGR-PURGED-AUDIT-COPY—315
Cause: The Manager process purged a backup copy of an audit trail file, probably to make room for a new copy.

Action: None. Informational.
Tokens:

`GGS-TKN-AUDIT-FILE`

GGS-EVT-MGR-AT-THRESHOLD—316
Cause: Manager detected that an audit trail file required by Extract is in danger of being deleted by TMF. The `PCT LEFT` indicates the amount of audit remaining before TMF purges data that Extract has not yet processed. For example, if the percent left is 20, 80 percent of audit remains to be processed.

Action: If Extract is down, start it. Otherwise, consider increasing priority for the Extract and Audserv processes.
Tokens:

`GGS-TKN-AUDIT-FILE`
`GGS-TKN-AT-PCT`

GGS-EVT-MGR-AT-DISK-THRESHOLD—317
Cause: Manager detected that audit on disk that is required by Extract is in danger of being deleted by TMF. The `PCT LEFT` figure indicates the amount of audit remaining before TMF purges data that Extract has not yet processed. For example, if the percentage left is 20, 80 percent of all audit currently on disk remains to be processed.

Action: If Extract is down, start it. Otherwise, consider increasing priority for the Extract and Audserv processes. Extract will request a tape restore if an audit dump is available.
Tokens:

`GGS-TKN-AUDIT-FILE`
`GGS-TKN-AT-DISK-PCT`

GGS-EVT-CREATE-LOG—318
Cause: Manager or GGSCI created a new log file for Logger.

Action: None required. Informational.
Tokens:

GGS-TKN-LOG-FILE

GGS-EVT-MGR-LOG-THRESHOLD—319
Cause: Manager is reporting that a Replicat or Extract process is falling a significant amount of log files behind the current file. If this problem persists, the Replicat or Extract process is in danger of losing data it needs to process since log data is eventually recycled. The message is reported only when LOGFILESBEHIND or LOGFILESBEHINDINFO is set.

Action: Boost the priority of the Replicat or Extract process that is falling behind to let it catch up. You may also want to increase the NUMFILES parameter in the LOGPARM file, then use GGSCI ALTER LOGGER to increase the amount of available log space.
Tokens:

GGS-TKN-LOG-FILE
GGS-TKN-FILES-BEHIND
GGS-TKN-GROUP

GGS-EVT-MGR-RESTARTED-PROCESS—320
Cause: Manager is restarting a Logger, Extract or Replicat process that abended, was stopped from TACL, or went down for some other re-tryable reason.

Action: None required. Informational.
Tokens:

GGS-TKN-GROUP

GGS-EVT-MGR-LAG-THRESHOLD—321
Cause: Generated by Manager whenever LAGINFO or LAGCRITICAL has been specified and the lag for the process exceeds the configured threshold. This message warns that a problem may exist in the system.

Action: Investigate possible performance problems (such as process running at low priority, or a faulty network).

GGS-EVT-REP-NEW-COLUMN—330
Cause: Generated when Replicat replicates an ADD COLUMN statement to the target database.

Action: None. Used to trace processing activity.

GGS-EVT-MGR-SLAVE—332
Cause: A Manager reported error originating in the slave Manager process.

Action: None. Used to trace processing activity.

GGS-EVT-MGR-CHILD—333
Cause: A Manager reported error originating in a child process.

Action: None. Used to trace processing activity.

GGS-EVT-CHILD-ADDEDEXISTING—340
Cause: When Manager restarts it puts out an informational message indicating that it found an existing Replicat, Extract, or synchronization process running and will track it.

Action: None. Used to trace processing activity.

GGS-EVT-CHILD-NOTADDED—341
Cause: A Manager reported error when a child process cannot be added to the child table.

Action: None. Used to trace processing activity.

GGS-EVT-NONSTOP—350
Cause: Errors related to NonStop processing such as start back up, takeover checkpoint errors, etc.

Action: None. Used to trace processing activity.

GGS-EVT-CPUUP—353
Cause: Manager is reporting CPU recovery messages to EMS/log file for reference.

Action: None. Information only.

GGS-EVT-SYNC-STARTED—360
Cause: Indicates Syncfile has started.

Action: None. Informational only.

GGS-EVT-SYNC-STOP-USER—361
Cause: Syncfile was stopped by the user, via GGSCI.

Action: None. Information only.

GGS-EVT-COLL-OPENING-FILE—370
Cause: Indicates that a Collector opened a file.

Action: None. Used to trace processing activity.

GGS-EVT-COLL-CLOSING-FILE—371
Cause: Indicates that a Collector closed a file.

Action: None. Used to trace processing activity.

GGS-EVT-COLL-TERMINATING—372
Cause: Indicates that a Collector is terminating normally (due to Extract stopping).

Action: None. Used to trace processing activity.

GGS-EVT-COLL-LISTENING—373
Cause: Indicates that a Collector has been started and is waiting for incoming requests.

Action: None. Used to trace processing activity.

GGS-EVT-REP-WAIT-FEVENT—380
Cause: Signals that Replicat is waiting on a file event such as `RENAME` or `DUP` to occur against a specific file. Replicat suspends processing until the event has completed or been cancelled through GGSCI.

Action: None. Used to trace processing activity.

GGS-EVT-REP-WAIT-COMPLETED—381
Cause: Signals that a file event on which Replicat was waiting completed, and that Replicat will resume processing other transactions.

Action: None. Used to trace processing activity.

GGS-EVT-OPENED-SOURCE-FILE—382
Cause: Issued by Extract when it opens a file that is not an Oracle GoldenGate trail or Logger trail file.

Action: None. Used to trace processing activity.

GGS-EVT-CLOSED-SOURCE-FILE—383
Cause: Issued by Extract when it closes a file that is not an Oracle GoldenGate trail.

Action: None. Used to trace processing activity.

GGS-EVT-GROUPLIST—384
Cause: Manager has filled its internal table of checkpoint groups and produced the error messaging:

```
*ERROR* group_list table limit reached.
```

Action: Stop and re-start Manager. If the problem continues, you have reached the maximum number of groups, which is 200.

GGS-EVT-TASK—385
Cause: Indicates there are messages from an Extract or Replicat task.

Action: None. Informational only.

GGS-EVT-CHKPT-REC-DELETED—386
Cause: Result from a GGSCI `CLEANUP` command.

Action: None. Informational only.

GGS-EVT--POS—387
Cause: Indicates if Extract and/or Replicat are started after a specified begin time. Message can be informational, if records are available to use for positioning, or a warning, if records are not available.

Action: None. Informational only.
Informational:

```
GGS INGO 387 Positioning for Begin time search, using \NODE.$VOL.SUBVOL.AA000001.
```

Warning:

```
GGS WARNING 387 No data found in /logtrail for Begin time search, using
\NODE.$VOL.SUBVOL.AA00009.
```

GGS-EVT-CHKPT-REC-CORRUPT—389
Cause: `RESTARTCHECKPOINTS` checkpoint metadata is no longer valid. See the error detail in the message for more detail.

Action: Correct the issue or delete the Extract group and re-add it.

OGG ERROR TACLCMD—392
Cause: Invalid or missing input or error on attempted execution of the `TACLCMD` statement. See the data returned with the error for more information.

Action: Correct the issue or remove the `TACLCMD` statement.

XR-FILE-ERR—500
Cause: This is an internal error code returned from the module that reads Oracle GoldenGate trails and does direct reads on user data files.

Action: XXXXX_NO_ACTION_or_"RECOVERY"_TEXT

XR-LOGIC-ERROR—514
Cause: Extract received an unexpected reply code from Replicat during direct load and generates the message:

```
Invalid reply code received.
```

Action: Examine the error message to determine the exact nature of the error.

GGS-EVT-LOGR-FILE-ERR—1000
Cause: Logger experienced a Guardian file error. Message text describes exact problem.

Action: Fix error and restart Logger. Resynchronization may be required.

GGS-EVT-LOGR-SHUTDOWN-RQST—1008
Cause: Logger received a shutdown request from GGSCI.

Action: None required. Informational.

GGS-EVT-LOGR-SHUTDOWN-COMPLETE—1009
Cause: Logger completed a shutdown request from GGSCI.

Action: None required. Informational.

GGS-EVT-LOGR-ABEND—1010
Cause: Logger abended for reasons provided in another error message.

Action: Manager may restart Logger in this situation depending on the reason Logger abended. Recovery depends on the conditions that caused Logger to abend.

GGS-EVT-LOGR-STARTED—1011
Cause: Logger startup message.

Action: None required. Informational.

WRN-FILTER-NOT-PASSED—1280
Cause: Replicat issues a warning that a filter did not pass the evaluation test for a record. This error may also generate other associated messages.

Action: Examine the `FILTER` clauses in the Replicat parameter file. Also examine the `REPERROR` parameter.

GGS-EVT-POOL-CREATE-ERR—1500
Cause: Memory allocation problems.

Action: Contact Oracle GoldenGate Support.

GGS-EVT-LOGR-LOGCONF-ERR—1600
Cause: The `LOGCONF` file may have been purged from configuration.

Action: Contact Oracle Support.

GGS-EVT-LOGR-TRAIL-OPEN—1800
Cause: Logger opened the next log file in the log trail.

Action: None required. Informational.

GGS-EVT-LOGR-TRAIL-SWITCH—1801
Cause: Logger will open the next log file in the log trail by request.

Action: None required. Informational.

GGS-EVT-LOGR-LOGGING-DISABLED—1802
Cause: Logger disabled logging by request.

Action: None required. Informational.

GGS-EVT-LOGR-LOGGING-ENABLED—1803
Cause: Logger enabled logging by request.

Action: None required. Informational.

GGS-EVT-INTERCEPT-NOTBOUND—2000
Cause: An application that opens a non-audited file for update access does not have the Oracle GoldenGate intercept library bound to it.

Action: Bind the Oracle GoldenGate intercept library to the application.

ECONNABORTED—4119
Cause: A connection was closed by the internal software on your host computer.

Action: Close the socket. Reestablish the connection using the socket, bind, and connect calls. If the problem persists, contact your Tandem representative.

ECONNREFUSED—4127
Cause: The connect call failed because the remote host rejected the connection request. This error usually results from an attempt to connect to a service that is inactive on the remote host.

Action: Start the server on the remote host. Close the local socket. Reestablish the connection using the socket, bind, and connect calls.

ECONNRESET—4120
Cause: The connect call failed because the peer process reset the connection before the operation completed.

Action: Close the local socket. Reestablish the connection using the socket, bind, and connect calls.

ENETDOWN—4116
Cause: The call failed because the operation encountered a down network.

Action: Contact the network Manager.

ENETRESET—4118
Cause: The call failed, and all connections to the specified remote host were closed. The network dropped connection because of reset. The host you were connected to crashed and rebooted.

Action: Close the sockets using the close call. Reestablish the connections using the socket, bind, connect, and accept calls; then retry the call.

ENOTCONN—4123
Cause: The call failed because the specified socket was not connected.

Action: Ensure that the socket is connected, then retry the operation.

EPIPE—4032
Cause: The call failed because a write or send call was attempted on a local socket that had been previously closed with the shutdown call.

Action: Reestablish the connection using the socket, bind, and connect calls. Retry the write or send call.

ESHUTDOWN—4124
Cause: The call failed because the specified socket was already shut down.

Action: Reopen the remote socket using the open, bind, and accept calls. Re-establish the connection, if desired, with a call to `connect` or `connect_nw`.

ETIMEDOUT—4126
Cause: The connection timed out before the operation completed. The call failed.

Action: Close the local socket. Rebuild the local socket using the socket and bind calls. Call `connect` or `connect_nw` to reestablish the connection.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z