Oracle® GoldenGate for Mainframe

Reference Guide

Version 10.0

October 2009



Oracle® GoldenGate for Mainframe Reference Guide, version 10.4

Copyright © 1995, 2009 Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface **About the GoldenGate Guides** 15

New in Version 10.0 16

Chapter 1 **GGSCI Commands** 20

Commands summary 20

Manager commands 21

INFO MANAGER 22

REFRESH MANAGER 22

SEND MANAGER 22

START MANAGER 23

STATUS MANAGER 23

STOP MANAGER 24

Extract commands 24

ADD EXTRACT 24

ALTER EXTRACT 28

CLEANUP EXTRACT 29

DELETE EXTRACT 29

INFO EXTRACT 30

KILL EXTRACT 31

LAG EXTRACT 31

SEND EXTRACT 32

START EXTRACT 36

STATUS EXTRACT 36

STOP EXTRACT 37

Replicat commands 37

ADD REPLICAT 37

ALTER REPLICAT 41

CLEANUP REPLICAT 41

DELETE REPLICAT 42

INFO REPLICAT 42

KILL REPLICAT 43

LAG REPLICAT 43

SEND REPLICAT 44
START REPLICAT 47
STATUS REPLICAT 48
STOP REPLICAT 48
ER commands 49
Logger commands 50
ADD LOGGER 50
ALTER LOGGER 51
DELETE LOGGER 51
INFO LOGGER 52
SEND LOGGER 53
START LOGGER 55
STATUS LOGGER 55
STOP LOGGER 55
Trail commands 56
ADD EXTTRAIL 56
ADD RMTTRAIL 58
ALTER EXTTRAIL 59
ALTER RMTTRAIL 59
DELETE EXTTRAIL 59
DELETE RMTTRAIL 60
INFO EXTTRAIL 60
INFO RMTTRAIL 60
Database commands 61
CAPTURE TABLEDEFS 61
ENCRYPT PASSWORD 62
INFO DDLDEFS 63
INFO FILES 63
Audit trail commands 64
ADD ATCONFIG 64
ALTER ATCONFIG 65
DELETE ATCONFIG 65
INFO ATCONFIG 66
STATUS AUDITTRAIL 66
Remote checkpoint commands 66
ADD REMOTECHKPT 67
DELETE REMOTECHKPT 67
INFO REMOTECHKPT 67

TMF commands 67
REFRESHTMFINFO 67
TMFDUMPAGE 68
TMFDUMPINFO 68
TMFDUMPTABLEENTRIES 68
TMFREFRESHINTERVAL 68
TMFTRAILINFO 69
Coordinator commands 69
ADD COORDINATOR 69
ALTER COORDINATOR 70
DELETE COORDINATOR 71
INFO COORDINATOR 71
SEND COORDINATOR 72
START COORDINATOR 73
STATUS COORDINATOR 74
STOP COORDINATOR 74
Process commands 75
SEND PROCESS 75
Marker commands 75
ADD MARKER 76
INFO MARKER 78
Programs commands 79
BIND PROGRAMS 79
INFO PROGRAMS 80
LINK PROGRAMS 81
Report commands 82
SEND REPORT 82
VIEW REPORT 84
Syncfile commands 84
ADD SYNCFILE 84
ALTER SYNCFILE 85
DELETE SYNCFILE 86
INFO SYNCFILE 86
KILL SYNCFILE 86
START SYNCFILE 86
STATUS SYNCFILE 87
STOP SYNCFILE 87
Miscellaneous commands 87
! command 87

ENV . . . 88
FC . . . 88
HELP . . . 90
HISTORY . . . 90
INFO ALL . . . 90
LOG . . . 91
LOG STOP . . . 91
OBEY . . . 91
EDIT PARAMS . . . 92
VIEW PARAMS . . . 92
SYSTEM . . . 92
VIEW GGSEVT . . . 93

Chapter 2 GoldenGate Parameters . . . 95

Parameter summaries . . . 95
GLOBALS parameters summary . . . 95
Manager parameters summary . . . 97
Logger parameters summary . . . 99
CHGNOTE parameters summary . . . 101
Extract parameters summary . . . 101
Replicat Parameters Summary . . . 109
Coordinator parameters summary . . . 116
Syncfile parameters summary . . . 116
Alphabetical reference . . . 117
ABORTDUPERRWINDOW . . . 117
ADD DEFINE . . . 118
ALLOCFILES . . . 119
ALTFILERESOLVE | NOALTFILERESOLVE . . . 120
ALTINPUT . . . 120
ASSUMETARGETDEFS . . . 122
AUDITING . . . 123
AUDITREPS | NOAUDITREPS . . . 123
AUDITRETRYDELAY . . . 123
AUDSERVCACHEBLOCKS . . . 123
AUDSERVCPU . . . 124
AUDSERVPARAM . . . 124
AUDSERVPREFIX . . . 126
AUDSERVPROCESS . . . 126
AUDSERVPROGRAM . . . 127

AUTORESTART 127
AUTOSTART 129
BACKUPCPU 130
BEGIN 130
BULKIOLOAD | NOBULKIOLOAD 131
CHECKINTERVAL 131
CHECKMINUTES 131
CHECKPARAMS 132
CHECKPOINTSECS 132
CHECKUNIQUEKEY | NOCHECKUIQUEKEY 132
CLEANUPSAVECOUNT 133
COBOLUSEREXIT 133
COLMATCH 133
COMMENT 135
COMPRESSDELETES | NOCOMPRESSDELETES 135
COORDINATOR 136
CPU 137
CUSEREXIT 137
DEBUGONSTACKCHECK 137
DECRYPTTRAIL 137
DICTIONARY 138
DISCARDFILE 138
DISCARDROLLOVER 139
DISKTHRESHOLD | NODISKTHRESHOLD 140
DISPLAYTRAILSWITCH | NODISPLAYTRAILSWITCH 140
DOWNCRITICAL 140
DOWNREPORT 141
DUP 141
DUPONLYAFTEREVENT 143
DUPPROCESS 144
DYNAMICPARTITIONS 144
DYNAMICPORTLIST 144
DYNAMICPORTREASSIGNDELAY 145
EMBEDDEDMACROS | NOEMBEDDEDMACROS 145
EMSLOG 145
ENCRYPTTRAIL | NOENCRYPTTRAIL 146
END 146
ENTRYSEQUPDATES 147
EOFDELAY | EOFDELAYCSECS 148

ERREPLYTIMEOUT 148
ERROR59ROLLOVER 149
ETNEWFORMAT | ETOLDFORMAT 149
EVENT 149
EXCLUDEFILE 150
EXCLUDEGGSTRANSRECS 151
EXCLUDESUFFIXCHAR 152
EXPANDDDL 152
EXTFILE 153
EXTRACT 155
EXTTRAIL 155
FASTIO 156
FASTPOSITION | NOFASTPOSITION 156
FASTREADS | NOFASTREADS 156
FETCHCOMPS | FETCHLASTIMAGE 156
FILE 157
FILE | TABLE 162
FILEAGEDAYS 173
FILEEXCLUDE 174
FILEOPWARNING 174
FILERESOLVE 175
FILLSHORTRECS | NOFILLSHORTRECS 176
FILTERVIEW | NOFILTERVIEW 176
FLUSHCHECKPOINT | NOFLUSHCHECKPOINT 176
FLUSHSECS | FLUSHCSECS 176
FORCESTOPDELAY 177
FORCEUSESYSKEY | NOFORCEUSESYSKEY 177
FORMATASCII | NOFORMATASCII 178
FORMATLOAD 181
FORMATLOCAL 182
FORMATSQL | NOFORMATSQL 182
FUNCTIONSTACKSIZE 183
GETALTKEYS | IGNOREALTKEYS 183
GETAPPLOPS | IGNOREAPPLOPS 184
GETAUXTRAILS | IGNOREAUXTRAILS 184
GETCOMPS | IGNORECOMPS 184
GETCREATES | IGNORECREATES 185
GETDEFAULTS 185
GETDELETES | IGNOREDELETES 186

GETENV 187
GETFILEOPS | IGNOREFILEOPS 187
GETINSERTS | IGNOREINSERTS 188
GETMARKERS | IGNOREMARKERS 188
GETNETCHANGES | IGNORENETCHANGES 188
GETNEWCOLUMNS | IGNORENEWCOLUMNS 189
GETPARTONLYPURGEDATAS | IGNOREPARTONLYPURGEDATAS 189
GETPURGES | IGNOREPURGES 190
GETPURGEDATAS | IGNOREPURGEDATAS 190
GETRENAMES | IGNORERENAMES 190
GETREPLICATES | IGNOREREPLICATES 190
GETROLLBACKS | IGNOREROLLBACKS 191
GETUPDATEAFTERS | IGNOREUPDATEAFTERS 191
GETUPDATEBEFORES | IGNOREUPDATEBEFORES 192
GETUPDATES | IGNOREUPDATES 192
GROUPTRANSOPS 192
HANDLECOLLISIONS | NOHANDLECOLLISIONS 193
HEARTBEAT 194
HOMETERM 194
HOMETERMMESSAGES 195
HOST 195
IGNOREPARAMERROR 196
IGNORETMFDUMPS 196
INCLUDE 196
INITAUXPOSITION | NOINITAUXPOSITION 197
INSERTALLRECORDS | NOINSERTALLRECORDS 197
INSERTDELETES | NOINSERTDELETES 198
INSERTMISSINGUPDATES | NOINSERTMISSINGUPDATES 199
INSERTUPDATES | NOINSERTUPDATES 199
IPINTERFACE 199
LAGCRITICAL 200
LAGINFO 200
LAGREPORT 201
LAGSTATS 201
LIMITRECS 202
LIST | NOLIST 202
LOG 203
LOGFILESBEHIND | LOGFILESBEHINDINFO 204
LOGFILEOPENS 205

LOGGERFILENUM 205
LOGGERFLUSHRECS 206
LOGGERFLUSHSECS | LOGGERFLUSHCSECS 206
LOGGERTIMEOUTSECS 207
LOGGGSCICOMMANDS 207
LOGON 207
MACRO 208
MACROCHAR 210
MANAGERREQUIRED 210
MAP 210
MAPEXCLUDE 228
MAXABENDRESTARTS 228
MAXDISCARDRECS 229
MAXETCHECKPOINTSECS 229
MAXTRANSMEM 229
MAXTRANSOPS 230
MAXWILDCARDENTRIES 230
NETWORKCHECKPOINTS 231
NOTSTOPPABLE 231
NUMEXTRACTS 231
NUMFILES 232
OBEY 232
OLDGROUPNAMING 232
OMITAUDITGAPCHECK 233
OPENTIMEOUT | OPENTIMEOUTMINUTES 233
OPENWARNINGS 233
OVERRIDEDUPS | NOOVERRIDEDUPS 234
PARTMAP 234
PARAMCHECK 235
PASSTHRU | NOPASSTHRU 235
PORT 236
POSITIONFIRSTRECORD 236
PRIORITY 236
PURGEDATAALTFILES | NOPURGEDATAALTFILE 237
PURGEOLDEXTRACTS 237
PURGEOLDTASKS 239
PURGERESTORE | NOPURGERESTORE 240
READER 240
READTHRULOCKS | NOREADTHRULOCKS 241

RECEIVEQWARN . . . 241
RENAMEBUMPDELAY . . . 241
REPERROR . . . 242
REPLACEBADCHAR . . . 245
REPLACEBADNUM . . . 246
REPLICAT . . . 246
REPNEWCOLUMNS | NOREPNEWCOLUMNS . . . 247
REPORT . . . 247
REPORTCOUNT . . . 248
REPORTFILEEXTENTS . . . 248
REPORTROLLOVER . . . 249
REPORTTMFEXCEPTIONS . . . 249
REPSQLLOG . . . 250
RESTARTCOLLISIONS | NORESTARTCOLLISIONS . . . 251
RESTARTINTERVAL . . . 251
RESTORE | NORESTORE . . . 251
RETRYDELAY . . . 251
RETRYERR . . . 252
REVERSEWINDOWSECS | REVERSEWINDOWCSECS . . . 252
RMTBATCH . . . 253
RMTFILE . . . 258
RMTHOST . . . 259
RMTHOSTALT . . . 261
RMTTASK . . . 262
RMTTRAIL . . . 263
ROLLOVER . . . 263
SETENV . . . 264
SHORTREADDELAY . . . 265
SOURCEDEFS . . . 265
SOURCEISFILE | SOURCEISTABLE . . . 266
SPECIALRUN . . . 267
STATOPTIONS . . . 268
SUPPRESSALTERMESSAGES | NOSUPPRESSALTERMESSAGES . . . 269
SUPPRESSFILEOPMESSAGES | NOSUPPRESSFILEOPMESSAGES . . . 269
SUPPRESSMARKERMESSAGES . . . 269
SWAPVOL . . . 270
SYNCFILE . . . 270
SYSKEYCONVERT . . . 270
TABLE . . . 271

TABLEEXCLUDE 271
TALUSEREXIT 272
TCPFLUSHBYTES 272
TCPIPPROCESSNAME 272
TCPIPSWITCHERRS 273
TCPSOURCETIMER | NOTCPSOURCETIMER 273
THRESHOLD | NOTHRESHOLD 273
TMFDUMPAGE 274
TMFDUMPTABLEENTRIES 274
TMFEXCEPTIONS | NOTMFEXCEPTIONS 274
TMFREFRESHINTERVAL 275
TMFTRAILTRACE 275
TRACE 275
TRACEALLOPENS 276
TRACECLOSES 276
TRACEOPENS 276
TRACEPROCESSIOS 276
TRACESTATS 277
UPDATEDELETES | NOUPDATEDELETES 277
UPDATEINSERTS | NOUPDATEINSERTS 277
UPREPORT 277
VERBOSE 278
VERSIONERR 278
WAITFILEEVENT 279
WARNRATE 281
Y2KCENTURYADJUSTMENT | NOY2KCENTURYADJUSTMENT 281

Chapter 3 Collector Parameters 282

Collector parameters 282
QSAM configuration parameters 285

Chapter 4 Field Conversion Functions 288

Overview of functions 288
Function syntax 288
Function summaries 288
Alphabetical reference 291
BINARY 291
CASE 291
COLSTAT 292
COLTEST 292

COMPUTE . . . 293
DATE . . . 293
DATEDIFF . . . 296
DATENOW . . . 296
EVAL . . . 296
GETENV . . . 297
GETVAL . . . 305
HIGHVAL | LOWVAL . . . 306
IF . . . 307
NUMBIN . . . 308
NUMSTR . . . 308
RANGE . . . 308
STRCAT . . . 310
STRCMP . . . 310
STREQ . . . 310
STREXT . . . 311
STRFIND . . . 311
STRLEN . . . 312
STRLTRIM . . . 312
STRNCAT . . . 312
STRNCMP . . . 313
STRNUM . . . 313
STRRTRIM . . . 314
STRSUB . . . 314
STRTRIM . . . 315
STRUP . . . 315
TOKEN . . . 316
VALONEOF . . . 316

Chapter 5 DEFGEN Arguments . . . 317

Argument summary . . . 317
EXCLUDESYSTEM | INCLUDESYSTEM . . . 317
EXPANDDDL . . . 317
RECORDNAMEPROMPTING . . . 321

Chapter 6 DDLGEN Arguments . . . 322

Run-time arguments . . . 322
Template parameters . . . 322
OMITNULL | INCLUDENULL . . . 323
OMITNOTNULL | INCLUDENOTNULL . . . 323

OMITREDEFS | INCLUDEREDEFS . . . 323
OMITCOMMENTS | INCLUDECOMMENTS . . . 324
USESAMENULLS . . . 324

Chapter 7 MEASFLS and MEASRPT Parameters . . . 325

Parameter summaries . . . 325
MEASFLS parameters summary . . . 325
MEASRPTS parameters summary . . . 326
MEASFLS parameter reference . . . 328
DURATION . . . 328
FILE . . . 328
FILEMEGABYTES . . . 329
GETSYSTEMUTILS . . . 329
GETTMFDETAIL . . . 329
INTERVAL . . . 329
MAXFILESPERMEAS . . . 330
MEASFILES . . . 330
PURGEMEASFILES . . . 330
START . . . 330
STOP . . . 331
WILDCARD . . . 331
MEASRPT parameter reference . . . 332
AVGCOMPRESSEDBYTES . . . 332
DEFAULTEXCLUDES . . . 332
DURATION . . . 332
EXCLUDEALTKEYS . . . 333
EXCLUDEFILE . . . 333
EXCLUDEPROGRAM . . . 333
EXCLUDETANDEMFILES . . . 334
FILEDETAIL . . . 334
GETAUDITED . . . 334
GETENSCRIBE . . . 334
GETNONAUDITED . . . 334
GETSQL . . . 335
GETTMFDETAIL . . . 335
INCLUDEFILE . . . 335
INCLUDEPROGRAM . . . 335
INTERVAL . . . 336
LISTLIMIT . . . 336

MAXSTATPROGS . . . 337
MEASFH . . . 337
MEASFILES . . . 337
MAXSTATFILES . . . 337
MAXSTATPROGS . . . 338
PROGDETAIL . . . 338
PROGSTATS . . . 338
REPORTRATE . . . 338
RETENTION . . . 338
START . . . 339
STOP . . . 339
SUPPRESSZEROTOTALS . . . 339

Chapter 8 User Exit Functions . . . 340

Function summary . . . 340
EXIT_CALL_RESULT . . . 341
EXIT_CALL_TYPE . . . 341
EXIT_PARAMS . . . 342
EXIT_REC_BUF . . . 344
Calling environment functions . . . 344
COMPRESS_RECORD . . . 346
DECOMPRESS_RECORD . . . 348
FETCH_CURRENT_RECORD . . . 349
FETCH_CURRENT_RECORD_WITH_LOCK . . . 350
GET_ALTKEY_INFO . . . 352
GET_COLUMN_INDEX . . . 352
GET_COLUMN_NAME . . . 353
GET_ENV_VALUE . . . 354
GET_EXITPARAM_VALUE . . . 356
GET_EXTRBA . . . 356
GET_EXTSEQNO . . . 357
GET_FILENAME . . . 358
GET_FOPEN_NUM . . . 358
GET_NUM_COLUMNS . . . 359
GET_RECORD . . . 360
GET_RECORD_LENGTH . . . 362
GET_SYSKEY_LENGTH . . . 363
GET_TRANSACTION_IND . . . 364
GET_USER_TOKEN_VALUE . . . 364

GGS_EMSMESSAGE 366
GGS_REPORTMESSAGE 367
Chapter 9 Event Error and Warning Messages 368

PREFACE

About the GoldenGate Guides

The complete GoldenGate documentation set contains the following components:

Oracle® GoldenGate for Mainframe

- *GoldenGate for HP NonStop Administrator Guide:* Introduces GoldenGate components and explains how to plan for, configure, and implement GoldenGate on the NonStop platform.
- *Reference Guide:* Provides detailed information about GoldenGate parameters, commands, and functions for the NonStop platform.

Windows and UNIX platforms

- *Installation and Setup guides*: There is one such guide for each database that is supported by GoldenGate.
- *GoldenGate for Windows and UNIX Administrator Guide*: Introduces GoldenGate components and explains how to plan for, configure, and implement GoldenGate on the Windows and UNIX platforms.
- *Reference Guide*: Provides detailed information about GoldenGate parameters, commands, and functions for the Windows and UNIX platforms.
- *Troubleshooting and Tuning Guide:* Provides suggestions for improving the performance of GoldenGate in different situations, and provides solutions to common problems.

Conventions used in writing this manual

This manual uses the following style conventions.

- Parameter and command arguments are shown in upper case, for example:
  ```
  CHECKPARAMS
  ```
- File names, table names, and other names are shown in lower case unless they are case-sensitive to the operating system or software application they are associated with, for example:
  ```
  account_tab
  GLOBALS
  ```
- Variables are shown within < > characters, for example:
  ```
  <group name>
  ```

- When one of multiple mutually-exclusive arguments must be selected, the selection is enclosed within braces and separated with pipe characters, for example:

  ```
  VIEW PARAMS {MGR | <group> | <file name>}
  ```

- Optional arguments are enclosed within brackets, for example:

  ```
  CLEANUP EXTRACT <group name> [, SAVE <count>]
  ```

- When there are numerous multiple optional arguments, a placeholder such as `[<option>]` may be used, and the options are listed and described separately, for example:

  ```
  TRANLOGOPTIONS [<option>]
  ```

- When an argument is accepted more than once, an ellipsis character (...) is used, for example:

  ```
  PARAMS ([<requirement rule>] <param spec> [, <param spec>] [, ...])
  ```

New in Version 10.0

Parameters

- Prior to this version Audserv logged a message without abending when the security check omitted a file. The default behavior is now to log a message and abend (NOABENDONSECURITYCHECK). A new NOABENDONSECURITYCHECK option of AUDSERVPARAM allows this to be set back.

- The AUTOSTART parameter can now be used to trigger Manager to start Logger processes.

  The the syntax has changed to:

  ```
  AUTOSTART [<group type>] {<group name> | <process name>}
            [, ALLPROCESSES]
  ```

  Where: <group type> can now include LOGGER:

  <process name> is a required entry for the LOGGER <group type> used for the Logger process name in the format $xxnnn. Wildcards may be used for all or part of the name.

- AUTORESTART can now be used to trigger Manager to restart Extract and Replicat processes that have failed.

  The syntax is:

  ```
  AUTORESTART <process type> <group name>
  [, RETRIES <max retries>]
  [, WAITMINUTES <wait minutes>]
  [, RESETMINUTES <reset minutes>]
  ```

Where: RETRIES sets the maximum number of times that Manager should try to restart a process before aborting retry efforts. The default number of tries is 2.

WAITMINUTES is the amount of time to pause between discovering that a process has terminated abnormally and restarting the process. The default delay is 1 minute.

RESETMINUTES is the window of time during which retries are counted. The default is 20 minutes. After the time expires, the number of retries reverts to zero.

- CHECKPOINTSECS parameter can now be used to control the frequency of routine checkpoints in Replicat as well as Extract.

- The Syncfile DUP parameter has new options:
  - INCLUDEFILECODE to specify file codes to be included in the DUP
  - EXCLUDEFILECODE to specify file codes to be excluded from the DUP
  - NAME to assign a logical name to the DUP

- The new FILEAGEDAYS parameter specifies the number of days a file can be inactive before Extract, Replicat, or Audserv ages it off its file list. Audserv inherits this parameter from the Extract.

  The syntax is

  ```
  FILEAGEDAYS <days>
  ```

  Only files added with a wildcard match are eligible to be removed. Files cannot be aged off if they are designated in FILE and MAP statements that have filters, SQLEXEC statements, or column mappings that use column functions.

- The new Manager parameter IPINTERFACE can be used to restrict the Manager process to the interface specified by an input IP address or DNS name.

  ```
  IPINTERFACE {<ip address> | <dsn name>}
  ```

- STATOPTIONS is a new parameter with the option NOZEROSUPPRESS to trigger reporting zero counts for updates, inserts and delete operation in Extract and Replicat. For Replicat it also allows PROGSTATS to report the time spent creating a file, purging a file, or ending a transaction.

  The syntax is:

  ```
  STATOPTIONS [ZEROSUPPRESS | NOZEROSUPPRESS]
            [, PROGSTATS] (Replicat only)
  ```

  The default is to suppress zero counts and to not report on the Replicat timings.

- The new TALUSEREXIT parameter can be used to call custom TAL routines at various points in Extract and Replicat processing.

- SUPPRESSMARKERMESSAGES stops messages being generated when markers are processed.

For Extract and Replicat, the syntax is:

```
SUPPRESSMARKERMESSAGES
```

For GLOBALS, the syntax is:

```
SUPPRESSMARKERMESSAGES {YES | NO}
```

- The EXITPARAM "exitparam string" option of FILE and MAP is no longer limited in size.

GGSCI commands

- Version 10 Logger processes include a new HOTSWAP option for SEND LOGGER. HOTSWAP instructs Logger to change to the specified object file.

  The syntax is:

  ```
  SEND LOGGER [<process name>]
     [HOTSWAP <object name>]
  ```

  If a Logger is running at the time of migration from a GoldenGate version 10 or higher to another version, the MIGRATE process will prompt to see if there should be a HOTSWAP of the Logger object.

  Note that using HOTSWAP to change the Logger program must be coordinated with the manual update of BASELIB.

Column conversion functions

- The GGFILEHEADER option has been added to @GETENV to return attributes of the trail file header record sent to NonStop from open systems. Attributes that can be returned include the trail version, file sequence number and size, information on the first and last records, and information on the local environment.

Functionality

- User exits can now be written in TAL as well as in C or COBOL. The TAL syntax for each of the callback functions has been added to the User Exits chapter of the Reference Guide.

- A new function is available to user exits. GET_USER_TOKEN_VALUE allows the value of a token to be retrieved.

- The function, GET_EXITPARAM_VALUE, is now available to user exits. It retrieves the value of an oversized string (over 256 bytes) entered with the EXITPARAM option of a FILE or MAP statement.

- Primary key updates coming from open system databases into a NonStop Enscribe or SQL/MP database are supported as of Version 10. These will be replicated by deleting the target record and inserting a new record with the new primary key.

  Primary key updates are still not supported for entry-sequenced and queue files.

  To continue to discard primary key updates instead of processing them, contact *GoldenGate Technical Support*.

- Version 10 includes support for HP Blades.
- The default location where GoldenGate libraries, such as BASELIB and GGSLIB, look for the AUDCFG segment file can now be assigned when running BUILDMAC or NLDLIB.
- A new library, PCREATE, intercepts a C runtime operating system function to allow the creation of high pin processes.

CHAPTER 1

GGSCI Commands

The GoldenGate Command Interface (GGSCI) is the command line interface that allows you to interact with GoldenGate components. This chapter describes the purpose and syntax for all GoldenGate commands.

Commands summary

The following summarizes the functions that you can control with GoldenGate commands.

Command Group	Purpose
Manager commands	Start and stop the Manager program, refresh it to execute parameter changes, and determine whether or not 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 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 or not audit trail files are still required.
Remote checkpoint commands	Establish remote checkpoints that Manager checks before purging data that is used by Replicat processes.

Command Group	Purpose
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 GoldenGate.

Manager commands

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

- Refresh, to pick up changes to the MGRPARM parameter file.
- Determine whether or not the Manager process is running.
- 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” on page 55 in the *GoldenGate for HP NonStop Administrator Guide*.

INFO MANAGER

Use INFO MANAGER to determine whether or not the Manager process is running. If Manager is running, the port number is displayed.

Syntax `INFO MANAGER`

Example `Manager process $GGMGR is running (IP port 18819).`

REFRESH MANAGER

Use REFRESH MANAGER when you have changed the MGRPARM parameter file. When you execute REFRESH MANAGER, the parameter file is updated and GGSCI notifies you that the update has occurred.

NOTE REFRESH MANAGER does not refresh the PORT and TCPIPPROCESSNAME parameters. If changes are made to these parameters, stop and restart the Manager process.

Syntax `REFRESH MANAGER`

SEND MANAGER

SEND MANAGER communicates with the current Manager process.

Syntax
```
SEND MANAGER
{CHILDSTATUS |
GETPORTINFO [DETAIL] |
GETPURGEOLDEXTRACTS |
KILL <process name>}
```

Option	Description
`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, whether or not they are in use, as in `SEND MANAGER, GETPORTINFO DETAIL`
`GETPURGEOLDEXTRACTS`	Displays information about trail maintenance rules that are set with the PURGEOLDEXTRACTS parameter in the Manager parameter file. For more informaton about PURGEOLDEXTRACTS, see page 237.
`KILL <process name>`	Stops a process that was previously created by Manager. Manager returns an error if the process is not one it created.

Example 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         2007/08/16 14:39:07
```

Example The following report results from the command SEND MANAGER GETPURGEOLDEXTRACTS.

```
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

Starts the Manager process.

Syntax

```
START MANAGER
[, CPU <primary cpu>]
[, BACKUPCPU <backup cpu>]
[, PRI <priority>]
```

Option	Description
`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 this command to determine whether or not 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 10.123.045.678 port 7670)
```

STOP MANAGER

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

Syntax

```
STOP MANAGER [!]
```

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

Extract commands

Use Extract commands to establish an Extract group so that audit trails are processed accurately from run to run. After the initial run, checkpoints provide a starting point for subsequent runs. Extract commands also provide run history information and display the audit trails required for a given Extract group.

Process names, parameter files, and report files take system-assigned default values. GoldenGate Software recommends using the default names. If your installation requires different names see “Changing default component names” on page 55 in the *GoldenGate for HP NonStop Administrator Guide*.

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 the argument table on page 25.

Syntax

```
ADD EXTRACT <group name>
{
[, BEGIN <time> |, AUDSEQNO <seq num>, AUDRBA <rba>] |
[, EXTTRAILSOURCE <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 <params file name>]
[, REPORT <report name>]
[, DESC "<text>"]
```

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

- Begins at midnight on May 1, 2006
- Runs in CPU 9, priority 170 and assigns a backup CPU in case the primary one crashes.

```
ADD EXTRACT DISTRIB, BEGIN 2006-05-01 00:00, CPU 9, BACKUPCPU 7, PRI 170
```

Table 1 ADD EXTRACT options summary

Argument	Description
`<group name>`	The group name.

`BEGIN <time> |`
`, AUDSEQNO <seq num>`
`, AUDRBA <rba>`
	The default source for ADD EXTRACT is the TMF audit trail. You can specify a starting point in an audit trail or a GoldenGate trail. See "Specifying a starting point" on page 26.
	
	For information on other data sources see "Specifying the data source" on page 26.

`CPU <cpu>`
`BACKUPCPU <cpu>`
`PRI <priority>`
	See additional information on assigning CPUs on page 27.

`DESC "<text>"`	See "Describing the group" on page 28.
`PARAMS <params file name>` `REPORT <report name>`	See "Specifying an alternative parameter or report file" on page 27.
`PROCESS <process name>`	See "Specifying an alternative process" on page 27.
`PROGRAM <program name>`	The name of the object file to execute. See "Executing user exits" on page 28.

Specifying the data source

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

- A local GoldenGate trail.
- A 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 GoldenGate trail

A local GoldenGate trail uses EXTTRAILSOURCE <trail name>. The following example identifies the data source as a local GoldenGate trail, and specifies a sequence number in the trail at which to begin extracting data.

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

Using the Logger trail

Specify the LOGTRAILSOURCE <trail name> option as in:

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

Using a file

When the source is an entry-sequenced or ACI file, specify the FILETYPE <file type> <file name> option, as in:

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

- For <file type>, enter one of: ENTRY, ACITLF, ACIPTLF, ACITLFX, or ACIPTLFX.
- Additionally, when capturing directly from a sequence of files, include the ALTINPUT and RANGE parameters in the Extract parameter file.

For one-time processing

For initial synchronization or other one-time tasks, specify SOURCEISTABLE (or SOURCEISFILE for an Enscribe file).

```
ADD EXTRACT GROUP1, SOURCEISTABLE
```

SOURCEISTABLE does not maintain checkpoints. When you configure Extract for a task, you must include a corresponding SOURCEISTABLE parameter in the Extract parameter file.

Specifying a starting point

You can specify a starting point within an audit trail or within a local GoldenGate trail. Typically, you would specify a starting point using BEGIN with a date and time. Using BEGIN is the preferred method.

Argument	Description
`BEGIN <time>`	Determines when Extract begins processing data in the audit trail. The <time> options are: NOW, or date/time as yyyy-mm-dd [hh:mi:[ss[.cccccc]]].
`AUDSEQNO <seq num>`	Identifies a specific sequence number for the TMF audit trail file at which to begin extracting data
`AUDRBA <rba>`	Specifies that processing begin at the specified relative byte address.
`EXTSEQNO <seq num>`	Identifies a specific sequence number for the GoldenGate trail file 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.

Argument	Description
`CPU <cpu>`	The primary CPU on which Extract runs. The default is the CPU on which Manager runs.
`BACKUPCPU <cpu>`	An alternate 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 You can assign both the primary and backup CPUs and a priority, as in:

```
ADD EXTRACT DISTRIB, BEGIN 2006-05-01 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. 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 2006-05-01 00:00, PROCESS $GGE07`

Specifying an alternative parameter or report file

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 page 95. Also see the *GoldenGate for HP NonStop Administrator Guide*.

- The default parameter file name is <GGS volume>.GGSPARM.<group name>, were <group name> represents a group, such as FINANCE.
- The default report file name is <GGS volume>.GGSRPT.<rpt name>, were <rpt name> represents the report file name, such as FINANCE. 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:

Argument	Description
`PARAMS <params 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 2006-05-01 00:00, PARAMS $DATA01.NEWPARM.FINANCE
ADD EXTRACT FINANCE, BEGIN 2006-05-01 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 execute 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 the *GoldenGate for HP NonStop Administrator Guide*.

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

Primarily, use ALTER EXTRACT to change attributes of the CPU, PRIORITY or BACKUPCPU options. ALTER EXTRACT accepts the same options as ADD EXTRACT. You can use ALTER EXTRACT to change the attribute of any option that you may have specified with ADD EXTRACT.

NOTE 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 EXTTRAILSOURCE or LOGTRAILSOURCE settings with ALTER EXTRACT, but GoldenGate recommends deleting and re-adding the group instead.

Syntax

```
ALTER EXTRACT <group name>
[, ETROLLOVER]
[, ETPURGE]
[, <option>]
```

Argument	Description
`<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. 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 all of the run history records for the FINANCE group except the last records, which it keeps.

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>]
```

Argument	Description
`<group name>`	An Extract group name or wildcard specification, such as * or FIN*.
`SAVE <count>`	Save the last <count> 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, GoldenGate deletes both the group and the metadata that controls the group's trail. By default it retains all the files currently contained within the trail. If you wish to delete captured data from your trail, you must use the exclamation point (!) in the DELETE EXTRACT statement or manually purge the files.

Syntax

```
DELETE EXTRACT <group name> [!]
```

Argument	Description
`<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.
- Obtain information about tasks.

Syntax

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

Argument	Description
`<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.

Argument	Description
`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 GoldenGate reports lag, see the *GoldenGate for HP NonStop Administrator Guide*.
`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 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>
```

Argument	Description
`<group name>`	The group name. You may 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 GoldenGate reports lag, see the *GoldenGate for HP NonStop Administrator Guide*.

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>}
```

Argument	Description
`<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 the options table on page 33.

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
}
```

Argument	Description
`<group name>`	A running Extract group. If the group specified is not running, an error is returned.

Example

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

Table 2 SEND EXTRACT options summary

Option	Operation
AUDITEND STATUS REPORT GETTCPSTATS RESETTCPSTATS	See "Obtaining process reports" on page 33.
ARCLOSECATALOG GETEXTARSTATS RESETEXTARSTATS GETARSTATS RESETARSTATS GETTRANSINFO GETARPROCESS GETARPARAMS GETARFILELIST GETARFILESTATS GETAREXCLUDELIST CLEAREXCLUDELIST	See "Managing the Audserv program" on page 34.
ROLLREPORT	See "Opening a new report file" on page 34.
ROLLOVER	See "Rollover GoldenGate trails" on page 35.
LAGSTATS <option>	See "Obtaining lag reports" on page 35.
FORCESTOP | STOP	See "Stopping the process" on page 35.
GETROLLBACKS | IGNOREROLLBACKS	See "Processing rollbacks" on page 36.

Obtaining process reports

You can generate reports for:

Report	Option	Description
End of audit trail	AUDITEND	Queries the Extract process to determine whether or not all records in the audit trails have been processed. This command indicates whether or not more Extract and Replicat activity must occur before a scheduled switch between databases. Until AUDITEND returns "All audit processed," more data needs to be processed before it can be assumed that secondary databases are synchronized.
Processing status	STATUS	Returns a detailed status of the processing state, including current position and activity.

Report	Option	Description
Processing statistics	REPORT	Generates an interim statistical report to the report file, including the number of inserts, updates, and deletes. Refer to page 82 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.
	RESETTCPSTATS	Resets the TCP/IP statistics so the next report displays fresh statistics.

Example For the following examples, the first 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.

Option	Description
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.

Option	Description
GETARPARAMS, [MAT | AUXnn]	Retrieves Audserv runtime 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 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

Option	Description
LAGSTATS <option>	Retrieves and optionally reports lag statistics. The options are the same as those for the LAGSTATS parameter. See additional LAGSTATS information page 201. 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:

Option	Description
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:

Option	Description
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>
```

Argument	Description
<group name>	The name of the Extract group. You may 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]
```

Argument	Description
<group name>	The name of the group. You may 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 or not the trails still exist. DETAIL is useful for determining whether or not audit needs to 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 |!]`

Argument	Description
`<group name>`	The name of the Extract group. You may 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 don't specify WAIT, GGSCI issues the next prompt immediately.
`ATEND`	Instructs Extract to stop when it reaches the end-of-file in the 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 a Logger trail, ATEND causes Extract to terminate when end-of-file is reached. If Extract is reading data from a GoldenGate trail instead of TMF audit trails, ATEND causes Extract to terminate when either end-of-file is reached or the Extract process writing to the trail has terminated, indicating that no more data is expected in the trail.
`!`	(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 the *GoldenGate for HP NonStop Administrator Guide*.

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 | EXTTRAIL <trail name> | LOGTRAIL <trail name>}
[, BEGIN <time> |, EXTSEQNO <seq num>, EXTRBA <rba>]
[, CPU <primary cpu>]
[, BACKUPCPU <backup cpu>]
[, PRI <priority>]
[, PROCESS <process name>]
[, PARAMS <params file name>]
[, REPORT <report name>]
[, DESC "<text>"]
[, PROGRAM <program name>]
```

Argument	Description
`<group name>`	Required. Up to 7 characters to designate some logical function of this Replicat group. You may 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, GoldenGate recommends constraining group names to 7 characters.
`SPECIALRUN | EXTTRAIL <trail name> | LOGTRAIL <trail name>`	Must be SPECIALRUN or one of the two record sources. A warning is issued if the specified record source does not exist. ◆ For SPECIALRUN see "Configuring initial data synchronization or other tasks" on page 40. ◆ Specify EXTTRAIL <trail name> when the source is a local trail. ◆ Specify LOGTRAIL <trail name> where the record source is a Logger trail.
`BEGIN <time>`	Determines when Replicat begins processing data, as in: `ADD REPLICAT ORDERS, EXTTRAIL \NY.$DATA2.EXTDAT.AA, BEGIN NOW, CPU 6, PRI 170` 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 num>, EXTRBA <rba>`	See "Specifying a starting point" on page 39.

`CPU <cpu>`
`BACKUPCPU <cpu>`
`PRI <priority>`
	Assign CPUs. For more information on assigning CPUs see page 39.

`DESC "<text>"`	See "Enter a Replicat group description" on page 40.

Argument	Description
`PARAMS <params file name>` `REPORT <report name>`	See "Specifying an alternative parameter or report file" on page 40.
`PROCESS <process name>`	See "Specifying an alternative Replicat process" on page 40.
`PROGRAM <program name>`	The object to be executed. See "Executing user exits" on page 41.

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, EXTTRAIL \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:

Option	Description
`BEGIN <time>`	Determines when Replicat begins processing data the audit trail. The <time> options are: NOW, or date/time as yyyy-mm-dd [hh:mi:[ss[.cccccc]]].
`EXTSEQNO <seq num>`	Identifies a specific sequence number in the local GoldenGate trail at which to begin extracting data. For example, if the EXTTRAIL 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 GoldenGate trail at the specified relative byte address.

Example
```
ADD REPLICAT ORDERS, EXTTRAIL $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:

Option	Description
`CPU <cpu>`	The processor on which Replicat will run. The default is the CPU on which Manager runs.
`BACKUPCPU <backup 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 NonStop documentation for more information.

Specifying an alternative Replicat process

The default process name is $GGRnn, where nn represents the sequence of the process. 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, EXTTRAIL $SYSTEM.GGSDAT.ET, PROCESS $GGR04
```

Specifying an alternative parameter or report file

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 page 95 for more detail on this parameter. Also see the *GoldenGate for HP NonStop Administrator Guide*.

- 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.<rpt name>, where <rpt name> represents the report file name, such as FINANCE. 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:

Option	Description
PARAMS <params 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,
EXTTRAIL $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, EXTTRAIL $SYSTEM.GGSDAT.ET,
DESC "T24 data pump for ATM transactions to IBM in Seattle"
```

Executing user exits

You can create and execute 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 the *GoldenGate for HP NonStop Administrator Guide*.

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>...]`

Argument	Description
`<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 EXTTRAIL 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 <group name> 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>]`

Argument	Description
`<group name>`	The group name. You may 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> [!]
```

Argument	Description
`<group name>`	The group name. You may use wildcards to specify a set of group names, such as, * or *FIN*.
`!`	(Exclamation point) Deletes each group without prompting the 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]
```

Argument	Description
`<group name>`	The group name. You may 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.

Argument	Description
`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>
```

Argument	Description
`<group name>`	The group name. You may 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>}
```

Argument	Description
`<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 Table 3.

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
}
```

Argument	Description
<group name>	A running Replicat group. If the group is not running, an error is returned.

Table 3 Options Summary

Option	Operation
FORCESTOP | STOP	See "Stopping the process" on page 47.
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" on page 46.

Option	Operation
`LAGSTATS <option> | LAGSNAPSHOT | LAGREPORTON | LAGREPORTOFF | LAGOFF`	See "Obtaining lag reports" on page 45.
`ROLLREPORT |`	See "Opening and closing the report file" on page 45.
`STATUS | REPORT`	See "Obtaining process reports" on page 45.

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 page 82 for detail on SEND REPORT options.

Opening and closing the report file

To close the current report file and open a new one, specify the ROLLREPORT option. ROLLREPORT renames the current file to <report file>1, 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.

Obtaining lag reports

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

Option	Description
`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.

Option	Description
`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.

Example 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 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 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.$DATA01.GGS.REPCTXT             2006/01/08 10:43:28     2    0
\LA.$DATA03.GGS.REPCTXT             2006/01/08 10:43:28     3    0
\SEA.$DATA01.GGS.REPCTXT            2006/01/08 10:43:28     4    0
```

Example 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.$DATA4.GGSDAT.ET000000, Current RBA       2280
Report at 2005-11-10 09:02:39 (Current settings)
Table/File                                            HANDLECOLLISIONS
MAP \LA.$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.

Option	Description
FORCESTOP	Instructs Replicat to rollback the pending transaction and stop gracefully
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 initiate a Replicat process. The START request is routed to Manager to start and monitor the process.

Syntax `START REPLICAT <group name>`

Argument	Description
<group name>	The group name. You may use wildcards to specify a set of group names, such as, * or *FIN*.

STATUS REPLICAT

Use STATUS REPLICAT to determine whether or not Replicat processes are running.

Syntax

```
STATUS REPLICAT <group name> [, DETAIL | TASKS | ALLPROCESSES]
```

Argument	Description
`<group name>`	The group name. You may 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 or not they are on disk or tape, and whether or not they still exist. DETAIL is useful for determining: ◆ Whether or not audit needs to 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 |!]
```

Argument	Description
`<group name>`	The group name. You may 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 don't specify WAIT, GGSCI issues the next prompt immediately.
`ATEND`	Instructs Replicat to terminate when it reaches the end-of-file in a trail. 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.

Argument	Description
!	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

GoldenGate allows you to 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> [, <options>]`

Argument	Description
`<command>`	Any one of the following: `INFO` Returns the processing status of both modules, including lag information. `KILL` Forces the processes to stop immediately. 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` Initiates the programs. The START command is routed to Manager, which starts and monitors the programs. `STATUS` Determines whether or not 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" on page 24 or "Replicat commands" on page 37.

Argument	Description
ER	Required. Informs 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
<options>	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" on page 24 or "Replicat commands" on page 37.

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 the *GoldenGate for HP NonStop Administrator Guide*.

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" on page 99 for information about the parameters you can enter into this file.

To bind GGSLIB to the application, see "Programs commands" on page 79.

Syntax

```
ADD LOGGER [, PARAMS <params file>]
```

Option	Description
PARAMS <params file>	Use the PARAMS option to indicate a different parameter file name.

ADD LOGGER performs the following:

- Validates the configuration parameters in 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 GoldenGate environment), see the *GoldenGate for HP NonStop Administrator Guide*.

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 <params file>]`

Option	Description
`PARAMS <params file>`	Use the PARAMS option to indicate a different parameter file name.

Configuration segment files are aged every time ALTER LOGGER is invoked (AUDCFG00, AUDCFG01, etc.). 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. Do this only when two conditions are met:

- The application has been stopped or suspended; and
- Downstream processes have completely processed all data in the existing log trails.

If these conditions do not exist, database integrity is at risk because updates to the same record can be processed out of original time sequence.

DELETE LOGGER

Use DELETE LOGGER to delete the internal 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, be sure that all of the data has been processed out of the trail.

Syntax `DELETE LOGGER [!]`

Option	Description
!	The exclamation point (!) deletes files from the associated log trail. If you omit this, the log trails associated with the process will need to be manually deleted 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 or not each Logger process in the group is running.
- CPU, BACKUPCPU and PRIORITY values for each process.
- Configured flushing and tracing parameters.
- 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]
```

Option	Description
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, etc.).
BRIEF	Limits the INFO LOGGER display to information about the SHOWLOGGED file name.

Option	Description
`SHOWLOGGED <file set>`	Allows you to determine which log process, if any, is capturing a particular file. Specify the file name or wildcard file set, whether or not the file is included, excluded or omitted for each Logger is displayed. If you specify SHOWLOGGED, you can also detect whether or not 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 (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 (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 (user set can be a single user, e.g. FINANCE.JOE, or a wildcard, e.g. SUPER.*).

SEND LOGGER

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

Syntax

```
SEND LOGGER
[, PROCESS < process name>]
[, ADJUSTPRIORITY | NOADJUSTPRIORITY]
[, ROLLOVER [,DETAIL]]
[, REFRESH]
[, PROCESSINFO [, DETAIL]]
[, LOGINFO [, DETAIL]]
[, FLUSHSTATS
   [, FILTERPROCESS <process name> |
      FILTERPROGRAM <program name> |
      FILTERLIBRARY <library name>] |
   [, RESET]]
[, GETSTATS
   [, FILTERPROCESS <process name> |
      FILTERPROGRAM <program name> |
      FILTERLIBRARY <library name>] |
   [, RESET]]
[, HOTSWAP <object name>]
```

Option	Description
PROCESS <process name>	If PROCESS is specified, the command is sent to the named process. Otherwise, the command is sent to all Logger processes in the default Logger group ($GGL*).
ADJUSTPRIORITY | NOADJUSTPRIORITY	Use ADJUSTPRIORITY or NOADJUSTPRIORITY to determine how Logger adjusts its priority in relation to the sender priority. Logger checks at each 1 minute interval 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. Optionally, include DETAIL to report process details.
PROCESSINFO	Instructs Logger to return input and output statistics, lag, and other information to the screen. Optionally, include DETAIL to report process details.
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 <proc name> ◆ FILTERPROGRAM <prog name> ◆ FILTERLIBRARY <lib 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.

Option	Description
	You can specify one of the following filters per SEND LOGGER GETSTATS command. ◆ FILTERPROCESS <proc name> ◆ FILTERPROGRAM <prog name> ◆ FILTERLIBRARY <lib 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.
`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 needs to be performed in conjunction with 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 initiates the ones that are down.

Syntax

```
START LOGGER [, PROCESS <process set>]
```

Option	Description
`PROCESS <process set>`	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>]
```

Option	Description
`NAME <process name>`	To get the status of a particular process within the group, specify the process name (e.g. $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 or not you wish 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>] [!]
```

Option	Description
`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 GoldenGate trails without bringing down the associated Extract process.

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

ADD EXTTRAIL

Use ADD EXTTRAIL 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.

Using ADD EXTTRAIL options, you can perform a variety of operations that are summarized in Table 4.

Syntax

```
ADD EXTTRAIL <trail name>, EXTRACT <group name>
[, OWNER <group num, user num>] [, SECURE “<rwep>”]
[, EXTENTS (<primary, secondary, maximum>) | MEGABYTES <num megabytes>]
[, MAXFILES <num files>]
```

Argument	Description
`<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 EXTTRAIL <trail name> parameter.

Argument	Description
EXTRACT <group name>	Specifies the group to which the EXTTRAIL is bound. Only one group can write to an associated trail.

Example This example adds a trail that is:

- Owned by user 100, 23 and can be read by anyone in the network in group 100.
- With the extents set to 10 for primary and secondary and 16 for the maximum.
- Set to a maximum of 20 trail files.

```
ADD EXTTRAIL $DATA1.EXTDAT.EX, EXTRACT FINANCE, MAXFILES 20, EXTENTS
(10,10,16), OWNER 100,23, SECURE "CUUU"
```

See *GoldenGate for HP NonStop Administrator Guide* for more information on managing trails.

Table 4. Options Summary

Option	Operation
EXTENTS (<primary, secondary, maximum>) | MEGABYTES <num megabytes>	See "Specifying file size" on page 57.
MAXFILES <num files>	See "Specifying a maximum number of files" on page 57.
OWNER <group num, user num> SECURE "<rwep>"	See "Specifying security" on page 58.

Specifying file size

You can control file size with one of the following options:

- Use the EXTENTS <primary, secondary, max> option to specify extent sizes for individual trails. Default extent sizes are 64, 128 and 512.

  ```
  ADD EXTTRAIL $DATA1.EXTDAT.EX, EXTRACT FINANCE, MAXFILES 20, EXTENTS
  (10,10,16)
  ```

- Use the MEGABYTES <num megabytes> 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 <num megabytes> to a large number, such as 1000 megabytes.

  ```
  ADD EXTTRAIL $DATA.GGSDAT.EX, EXTRACT FINANCE, MAXFILES 20, MEGABYTES 300
  ```

Specifying a maximum number of files

To specify a maximum number of files that can exist in a trail, specify the MAXFILES <num files> option, as follows:

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

The default for MAXFILES is 100.

Specifying security

You can specify security measures to restrict access to GoldenGate trails. If you don't specify security the defaults are assumed.

Option	Description
`OWNER <group num, user num>`	The NonStop group ID and user ID of the person who started the GGSCI process.
`SECURE "<rwep>"`	The default Guardian security attributes (read, write, execute, purge) of the person who started the GGSCI process.

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

ADD RMTTRAIL

ADD RMTTRAIL creates a remote 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 also be controlled 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 <num megabytes>]
```

Argument	Description
`<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 EXTTRAIL.
`EXTRACT <group name>`	The group to which the RMTTRAIL is bound. Only one group can output extracted data to each trail.
`MEGABYTES <num megabytes>`	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 <num megabytes> to a large number, such as 1000 megabytes.

Example The following example illustrates how to add three remote 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
```

ALTER EXTTRAIL

ALTER EXTTRAIL changes attributes for an existing EXTTRAIL.

Syntax `ALTER EXTTRAIL <trail name>, [, <options>]`

Argument	Description
`<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 EXTTRAIL takes the same options as ADD EXTTRAIL. For details on these options see ADD EXTTRAIL on see page 56.

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

ALTER RMTTRAIL

ALTER RMTTRAIL changes attributes for an existing RMTTRAIL.

Syntax `ALTER RMTTRAIL <trail name> [, MEGABYTES <num megabytes>]`

Argument	Description
`<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 EXTTRAIL.
`MEGABYTES <num megabytes>`	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 <num megabytes> to a large number, such as 1000 megabytes.

DELETE EXTTRAIL

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

Syntax `DELETE EXTTRAIL <trail name> [!]`

Argument	Description

`<trail name>`	The fully qualified trail name: $vol.subvol.<trail prefix>, or a wildcard specification, as in:
	`DELETE EXTTRAIL $DATA1.EXTDAT.AA`
	`DELETE EXTTRAIL *`

`!`	(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>`

Argument	Description
`<trail name>`	The fully qualified name of the remote 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 EXTTRAIL

INFO EXTTRAIL retrieves configuration information about the trail.

Syntax `INFO EXTTRAIL <trail name>`

Argument	Description

`<trail name>`	The fully qualified trail name: $vol.subvol.<trail prefix>, or a wildcard specification, as in:
	`INFO EXTTRAIL $DATA1.EXTDAT.AA`
	`INFO EXTTRAIL *`

INFO RMTTRAIL

INFO RMTTRAIL retrieves configuration information for the remote trail.

Syntax `INFO RMTTRAIL <trail name>`

Argument	Description
<trail name>	The name of the 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 EXTTRAIL 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 and subvol>]
[, RECORD <record name>]
[, OPTIONS <command line options>]
```

Argument	Description
<file name>	The fully qualified name of the file or table.
DICTIONARY <volume and subvol>	The volume and subvolume of the Enscribe dictionary. Required for Enscribe files.
RECORD <record name>	The name of the record defintion 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    (64)
    JTS_TEXT      CHAR
```

ENCRYPT PASSWORD

Use ENCRYPT PASSWORD to encrypt a login password for a 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 page 207).

For more information about GoldenGate security, see the *GoldenGate for HP NonStop Administrator Guide*.

Syntax

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

Argument	Description
`<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 GoldenGate and automatically decrypted on the target system. ◆ <keyname> Specifies an encryption key contained in the ENCKEYS lookup file. 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 <subvol>
[, DEFSONLY | RECSONLY]
```

Argument	Description
`<def name>`	The name of a DDL definition or record. <def name> can be specified multiple times to display multiple definitions or records. Wildcards are accepted.
`DICT <subvol>`	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]
```

Argument	Description
`<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.

Argument	Description
ENSCRIBE	Returns Enscribe files.
SQL	Returns NonStop SQL tables.
EXCLUDELASTDIGIT	Excludes file names which end in a digit (for the purpose of filtering 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 the *GoldenGate for HP NonStop Administrator Guide*.

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 GoldenGate support before using the DUP, DUPFILES, or PURGE options. These options do not represent best practices because they require storage that is not necessary if large enough audit trails are specified when TMF is configured.

Syntax

```
ADD ATCONFIG <atname>
[, ALTLOC <alt subvolume>]
[, DUP | NO DUP |
DUPFILES <num files> | NO DUPFILES |
PURGE | NO PURGE]
```

Argument	Description
<atname>	The audit trail designation, i.e. MAT, AUXnn. 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 alternate 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 alternate subvolume. When <num files> is reached on the alternate 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 alternate 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 <atname>
[, 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 executed, Manager will not manage the associated audit trails.

Syntax

```
DELETE ATCONFIG <atname>
```

Argument	Description
<atname>	The audit trail designation, i.e. MAT, AUXnn. 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 <atname>
```

Argument	Description
<atname>	The audit trail designation, i.e. MAT, AUXnn. 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 prior to processing — useful when an operator is not available while Extract processes are running.

Syntax

```
STATUS AUDITTRAIL <atname>
```

Argument	Description
<atname>	The audit trail designation, i.e. MAT, AUXnn. The audit trail can also be expressed as a wildcard.

Remote checkpoint commands

Use REMOTECHKPT when the Manager process is configured to perform trail maintenance using checkpoints and when the trail is being processed by processes 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 GoldenGate is installed, as in:

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

Syntax

```
ADD REMOTECHKPT <checkpoint file>
```

Argument	Description
`<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>
```

Argument	Description
`<checkpoint file>`	The checkpoint file to examine.

INFO REMOTECHKPT

INFO REMOTECHKPT lists all remote checkpoint references.

Syntax

```
INFO REMOTECHKPT <checkpoint file>
```

Argument	Description
`<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

TMFDUMPAGE limits information returned by TMFDUMPINFO to dumps that are created during the specified number of days.

Syntax `TMFDUMPAGE <num days>`

Option	Description
`<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>`

Option	Description
`<max dumps>`	The number of dump information 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>`

Argument	Description
`<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>"]
```

Argument	Description
`<group name>`	The group name.
`CPU <cpu>`	Specifies the primary CPU on which Coordinator runs. The default is the CPU on which Manager runs.
`BACKUPCPU <cpu>`	Specifies an alternate CPU on which Coordinator runs if the primary CPU becomes unavailable.

Argument	Description
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. 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>	Supplies an alternative parameter file name. Enter the fully qualified path name for the parameter file.
PROGRAM <program name>	This specifies the name of the program that Manager uses when starting the process. Typically this is not entered, and Manager uses the default $GGCnn name. The HOST parameter in the GLOBALS files is used to determine the location of the default program.
REPORT <report name>	Supplies an alternative report file name. The default report file name is <GGS volume>.GGSRPT.<rpt name>, where <rpt name> represents the report file name, such as FINANCE. 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 EXTTRAIL or to change the properties of an existing Coordinator group. You can use ALTER COORDINATOR to change the attribute of any option that you may have specified with ADD COORDINATOR.

Syntax
```
ALTER COORDINATOR <group name>
[EXTTRAIL <trail name>
{BEGIN <time> |, EXTSEQNO <seq number>, EXTRBA <rba>}]
[, <options>]
```

Argument	Description
<group name>	The group name.

Argument	Description
EXTTRAIL | EXTTRAILSOURCE {BEGIN <time> | , EXTSEQNO <seq number> , EXTRBA <rba>}	Specifies the starting point in the GoldenGate trail as a beginning time, transaction sequence, or relative byte address. The specified EXTTRAIL must match one of the trails defined in the Coordinator parameter file.
<options>	The ADD COODINATOR options can also 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 COORD <group name> removes the group and all checkpoints. Using the EXTTRAIL <trail name> option deletes only the trail checkpoints, but not the group.

Syntax

```
DELETE COORDINATOR <group name>
[EXTTRAIL <trail name>]
```

Argument	Description
<group name>	The group name. Using this option without the EXTTRAIL deletes the group and all trail checkpoints.
[EXTTRAIL <trail name>]	Deletes only the EXTTRAIL 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]
```

Argument	Description
<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).

Argument	Description
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 2006-12-01 15:59  Status RUNNING
Process  $GGC00       Checkpoint Lag: unknown
Checkpoints:
    Trail                 Time                          Seqno         RBA
\NY.$DATA1.GGSDAT.Z1    Updated at 2006-12-01 16:00:22.950722
                          2006-11-17 12:22:46.657637       0           0
                          2006-11-17 12:22:46.657637       0        1779
\LA.$DATA1.GGSDAT.EX    Updated at 2006-12-01 16:00:22.950722
                          2006-12-01 15:55:39.664490       0           0
                          2006-12-01 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}
```

Argument	Description
<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.

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

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

```
  279: 0 TransID 7926335489872297987  2006/11/17 12:22:47.068460
     Pending, Needed 2, Found 1, RefCount 1
     Bitmap 8000 0000 0000
  813: 0 TransID 7926335489872232451  2006/11/17 12:22:46.947382
     Pending, Needed 2, Found 1, RefCount 1
     Bitmap 8000 0000 0000
  825: 0 TransID 7926335489872363523  2006/11/17 12:22:47.317281
     Pending, Needed 2, Found 1, RefCount 1
     Bitmap 8000 0000 0000
  909: 0 TransID 7926335489872166915  2006/11/17 12:22:46.769463
     Pending, Needed 2, Found 1, RefCount 1
     Bitmap 8000 0000 0000
 1701: 0 TransID 7926335489872101379  2006/11/17 12:22:46.657637
     Pending, Needed 2, Found 1, RefCount 1
     Bitmap 8000 0000 0000
```

Example 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  2006/11/17 12:22:46.657637  0,1779
     Newest 7926335489872363523  2006/11/17 12:22:47.317281  0,4280
CurTransCount      5, LastRec 2006/12/01 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 2006/12/01 16:01:36.233081
 Records           11, Bytes          704, Transactions            0
Totals
Reader Requests           21, Records           25
Commit Requests            0
Force Commit 0
```

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

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

Syntax `START COORDINATOR <group name>`

Argument	Description
`<group name>`	The group name. You may 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]`

Argument	Description
`<group name>`	The group name. You may 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>`

Argument	Description
`<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 may 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}
```

Argument	Description
`<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, if you want 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, 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>]
```

Option	Description
`LOGGER <logger prefix> <freeform text>`	See page 76 for information on adding markers for TMF and non-TMF installations
`TACLCMD <program> <group name> <command>]`	See page 77 for information on executing TACL commands.
`GROUPCMD <program> <group name> <command>]`	See “Adding markers for a Replicat group” on page 78.

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, GoldenGate applies markers to the TMF audit trail.

Syntax

```
ADD MARKER [LOGGER <logger prefix>] [<freeform text>]
```

Option	Description
`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 $GGL ($GGL00, 01,...). 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 2006-07-30`

Executing 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 executed 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 executes, 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>
```

Argument	Description
`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 $GGL ($GGL00, 01,...). 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 executed. TACLCMD must include the following: ◆ <program> Either EXTRACT or REPLICAT. This determines which program executes the command. ◆ <group name> The group name to execute the command. You may specify a wildcard.

Argument	Description
	◆ <command> The TACL command to execute. 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 executed 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 a Replicat group

You can use markers to send a command to a Replicat group.

Syntax

```
ADD MARKER GROUPCMD REPLICAT <group name> <command>
```

Argument	Description
`REPLICAT`	Specifies a Replicat group.
`GROUPCMD <program> <group name> <command>`	The GROUPCMD keyword informs the process that a group command is to be executed. ◆ <program> Enter Replicat. ◆ <group name> Enter the 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.

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>]
```

Option	Description
`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:

Information Returned	Description
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 the GGSSRL or 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 <parameter file>]
[, REPORT <report file>]
[, GGSLIB <library file name>]
[, ERRORS <num errors>]
[, FORCEBIND]
[, NOLIBBIND]
[, CHANGELIB]
```

Option	Description
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, GoldenGate will run BIND PROGRAMS with the AXCEL option enabled.
PARAMS <parameter file>	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. <GGS volume>.GGSRPT.BIND is the default. Previous versions of the report file are aged to BIND00, BIND01, and so on.
GGSLIB <library file name>	Changes the name of the GGSLIB to bind with the application. The default is GGSLIB in the GoldenGate home subvolume.
ERRORS <library file name>	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 this, 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 set> [, BOUND | UNBOUND]`

Argument	Description
`<file set>`	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, GGSSRL or GGSDLL, to your application programs. You must bind the intercept library to capture non-audited database updates to Enscribe files. Once this is complete, GGSSRL/GGSDLL becomes the Guardian user library for specified programs (through the NLD -change libname command).

Syntax
```
LINK PROGRAMS
[, PARAMS <parameter file>]
[, REPORT <report file>]
[, GGSSRL <library file name>]
[, 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).

Option	Description
`PARAMS <parameter file>`	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 <GGS volume>.GGSRPT.LINK. Previous versions of the report file are aged to LINK00, LINK01, and so on.
`GGSSRL <library file name>`	Changes the name of the library to link with the application. The default name is GGSSRL for D45+/G06 applications or GGSDLL for H06/J06. These libraries are stored in the 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 GGSSRL. 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.

Table 5 Scrolling Commands

Command	Description
`<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 backwards 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 <program> <group name>,
REPORT [<time option> [RESET | FILE <name> | TABLE <name>]]
```

Argument	Description
`<program>`	Enter one of EXTRACT, REPLICAT, or SYNCFILE.
`<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>.

Argument	Description
	◆ <time option> Each of these reports statistics for a different time interval as listed below. REPORT is the default. 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. This is valid only for Replicat.

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

```
Report at 2006-02-27 12:43:07 (activity since 2006-02-27 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 The next sample Extract report uses the TOTALS option.

```
Report at 2006-02-27 12:44:15 counters for TOTALS since 2006-02-24 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 The following Replicat report uses the TRANSACTION option.

```
Report at 2006-02-28 06:51:15 counters for TRANS since 2006-02-14 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>`

Argument	Description
`<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 <parameter file>]
[, REPORT <report file>]
[, PROGRAM <program name>]
[, PROCESS <process name>]
[, CPU <primary cpu>]
[, BACKUPCPU <backup cpu>]
[, PRI <priority>]
[, DESC "<text>"]
```

Argument	Description
`<group name>`	A Syncfile group. You may use wildcards to specify a set of group names, such as * or *FIN*.
`PARAMS <parameter file>`	Specifies the Syncfile parameter file name.
`REPORT <report file>`	Specifies the name of the report to which Syncfile writes messages.
`PROCESS <process name>`	The process name. See "Specifying an alternative process" on page 27.
`PROGRAM <program name>`	The name of the object to be executed. See "Executing user exits" on page 28.
`CPU <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'll use ALTER SYNCFILE to change CPU, BACKUPCPU or PRIORITY options.

Syntax

```
ALTER SYNCFILE <group name> [, options>]
```

Argument	Description
`<group name>`	A Syncfile group. You may use wildcards to specify a set of group names, such as * or *FIN*.
`<options>`	ALTER SYNCFILE uses the same options as ADD SYNCFILE.

DELETE SYNCFILE

Use DELETE SYNCFILE to delete a Syncfile group.

Syntax `DELETE SYNCFILE <group name>`

Argument	Description
`<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 or not the process is running, and the date of the last time started.

Syntax `INFO SYNCFILE <group name> [, DETAIL]`

Argument	Description
`<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, as 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>`

Argument	Description
`<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. The START request is routed to Manager, which starts and monitors the Syncfile process.

Syntax `START SYNCFILE <group name>`

Argument	Description
`<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>`

Argument	Description
`<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>`

Argument	Description
`<group name>`	A Syncfile group. You can use a wildcard in the name.

Miscellaneous commands

The following commands control various other aspects of GoldenGate.

! command

Use the ! command to execute a previous GGSCI command without modifications. To modify a command before re-executing it, use the FC command. To display a list of previous commands, use the HISTORY command.

Issuing the ! command without arguments re-executes the most recently used command. By using options, you can re-execute a specific command by specifying its line number or a text substring.

Syntax `! [<number> | -<number> | <string>]`

Argument	Description
`<number>`	Re-executes the command on the specified GGSCI line. Each GGSCI command line is sequenced, beginning with 1 at the start of the session.
`-<number>`	Re-executes the command issued <number> lines before the current line.
`<string>`	Re-executes the last command that starts with the specified text string.

Example `! 9`

Example `! -3`

Example `! 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                        8.0.4.0 G06 2006/05/05
Prefix                         $GG
System                         \NY
Programs                       \NY.$DATA1.GGS8040
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 re-execute it. Previous commands are stored in the memory buffer and can be displayed by issuing the HISTORY command. The FC command is the same as standard NonStop FIX command functionality.

Issuing FC without arguments re-executes 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. To edit a command, 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 6 FC editor commands

Argument	Description

`i <text>`	Inserts text. For example:
	`GGSCI 24> fc 9`
	`GGSCI 24> send mgr`
	`GGSCI 24..  i childstatus`
	`GGSCI 24> send mgr childstatus`

`r <text>`	Replaces text. For example:
	`GGSCI 25> fc 9`
	`GGSCI 25> info mgr`
	`GGSCI 25..  rextract ggfin`
	`GGSCI 25> info extract ggfin`

`d`	Deletes a character. To delete multiple characters, enter a `d` for each one. For example:
	`GGSCI 26> fc 10`
	`GGSCI 26> info extract ggfin, detail`
	`GGSCI 26..  ddddddd`
	`GGSCI 26> info extract ggfin`

`<replacement text>`	Replaces the displayed command with the text that you enter on a one-for-one basis. For example:
	`GGSCI 26> fc 10`
	`GGSCI 26> info mgr`
	`GGSCI 26..  extract ggfin`
	`GGSCI 26> 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 forward slash (/) twice.

Syntax `FC [<number> | -<number> | <string>]`

Argument	Description
`<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.

Example `FC 9`

Example `FC -3`

Example `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>`

Argument	Description
`<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 find the saved volume for the user and then edit the GGSCIHST stored there.

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

Syntax `HISTORY [<number>]`

Argument	Description
`<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 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]`

Argument	Description
`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> | <process name> | <spooler>}`

Argument	Description
`<file>`	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 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>`

Argument	Description
`<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>`

Argument	Description
`<file name>`	Specify a parameter file name. If the file name is unqualified, EDIT assumes the parameter file resides in the GoldenGate home volume: <GGS volume>.GGSPARM.

VIEW PARAMS

VIEW PARAMS displays the named parameter file to your screen. Use the scrolling commands described in Table 7 to scroll through the file.

Syntax `VIEW PARAMS <file name>`

Argument	Description
`<file name>`	Specify a parameter file name. If the file name is unqualified, VIEW assumes the parameter file resides in the GoldenGate home volume: <GGS volume>.GGSPARM.

Table 7 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 backwards one page in file.
`q`	Quit display.
`h`	Help.

SYSTEM

The SYSTEM command enables you to manage 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 as 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 are used to identify and start GGSCI sessions on the remote systems.

Syntax

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

Argument	Description
`<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 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 NonStop's Event Management System (EMS), using VIEW GGSEVT is sometimes more convenient.

Syntax

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

Option	Description
`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 (e.g. 2006-03-31 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?

Option	Description
`<enter>`	Enter additional options for another GGSEVT display.
`q`	Exit GGSEVT and return to GGSCI.

CHAPTER 2

GoldenGate Parameters

This chapter describes GoldenGate parameters that are available to users for configuring, running, and managing GoldenGate processes.

Parameter summaries

This section summarizes GoldenGate parameters for HP NonStop, based on their functionality. An alphabetized reference follows on page 117.

GLOBALS parameters summary

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

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

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.

GLOBALS parameters: All

Parameter	Description
ADD DEFINE	Defines non-default locations for certain 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.

GLOBALS parameters: All (continued)

Parameter	Description
ERREPLYTIMEOUT	Sets the timeout value for GGSCI communication with GoldenGate components.
HEARTBEAT	Ensures that Extract and Logger can send data to Replicat.
HOMETERM	Sets up a global home terminal for GoldenGate processes.
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 or not user commands are written to the LOGGGS file.
MANAGERREQUIRED	Determines whether or not Extract and Replicat can run without a Manager process.
MAXWILDCARDENTRIES	Sets the initial allocation for wildcard entries.
OLDGROUPNAMING	Determines whether or not new Extract and Replicat group naming is enforced.
PARAMCHECK	Specifies whether or not 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 GoldenGate processes.

GLOBALS parameters: All (continued)

Parameter	Description
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 GoldenGate components such as Extract, Collector, and Replicat. Located in the GoldenGate subvolume, GGSCI expects the file to be named MGRPARM; this cannot be changed.

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	Allows you to store up to 6,000 TMF dump information entries.

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.

Manager parameters: Process management (continued)

Parameter	Description
RESTARTINTERVAL	Determines the period in which restart retries are counted before being reset.

Parameter	Description
CHECKMINUTES	Determines how often Manager cycles through its maintenance activities.
PURGEOLDEXTRACTS	Purges GoldenGate trails that are no longer needed according to specified criteria.
PURGEOLDTASKS	Purges Extract and Replicat tasks after a specified period of time.

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.

Manager parameters: Error and event management

Parameter	Description
DISKTHRESHOLD | NODISKTHRESHOLD	Generates an event message when an audit trail file that needs to 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.

Manager parameters: Error and event management (continued)

Parameter	Description
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 <GGS volume>.GGSPARM.LOGPARM. Enter parameters in the default file unless you have strong reasons to specify a different file name.

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.

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

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 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 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 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.

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 commands from a different parameter file.
SETENV	Sets a NonStop environment variable.

Extract parameters: Process management

Parameter	Description
ALLOCFILES	Controls the amount of incremental memory that is allocated when the amount of memory specfied with NUMFILES is reached.

Extract parameters: Process management (continued)

Parameter	Description
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.
DYNAMICPARTITIONS	Retrieves data from dynamically split partitions without restarting Extract.
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	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.

Extract parameters: Process management (continued)

Parameter	Description
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.
INITAUXPOSITION | NOINITAUXPOSITION	Controls the positioning of Audserv into the AUX trail.
LIMITRECS	Limits the number of records when capturing directly from a source table.
MAXWILDCARDENTRIES	Sets the initial allocation of wildcard entries.
NUMFILES	Controls the initial allocation of memory dedicated to storing information about files to be processed by GoldenGate.
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.
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.

Extract parameters: Security

Parameter	Description
DECRYPTTRAIL	Performs decryption when reading encrypted files in the GoldenGate trail.

Extract parameters: Security (continued)

Parameter	Description
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.

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 a 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.
ETNEWFORMAT | ETOLDFORMAT	Causes Extract to generate trails in a format that is compatible with Replicat version 6.0 and above or with earlier versions.
EXTFILE	Identifies the local GoldenGate trail as a flat file to which extracted data is output.
EXTTRAIL	Identifies the local GoldenGate trail to which extracted data is output.
FLUSHSECS | FLUSHCSECS	Determines the maximum time an extracted record remains buffered before being written to the GoldenGate trail.
NUMEXTRACTS	Allows Extract to exceed the default number of files in a GoldenGate trail.
OMITAUDITGAPCHECK	Enables Extract to continue processing even when a gap is detected between the oldest required audit trail and the current trail.
PURGEOLDEXTRACTS	Purges GoldenGate trail files when they are no longer required.
PURGERESTORE | NOPURGERESTORE	Purges audit files restored from tape immediately after use by Extract.

Extract parameters: File and trail management (continued)

Parameter	Description
RESTORE | NORESTORE	Controls whether or not audit dumps on tape are reloaded when the dumps are not available on disk.
ROLLOVER	Specifies conditions under which local and remote 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.

Extract parameters: Remote processing

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

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 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.

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.
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.

Extract parameters: Data selection (continued)

Parameter	Description
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.

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.

Extract parameters: Data formatting and conversion

Parameter	Description
FILLSHORTRECS | NOFILLSHORTRECS	Fills out records that are shorter than maximum length.

Extract parameters: Data formatting and conversion (continued)

Parameter	Description
FORMATASCII | NOFORMATASCII	Formats output in external ASCII format.
FORMATSQL | NOFORMATSQL	Formats extracted records into equivalent SQL INSERT, UPDATE, and DELETE syntax.
FORMATLOAD	Formats extracted records for compatibility with SQLCI LOAD utility.
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.

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.
MACRO	Creates a 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.

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.
ERROR59ROLLOVER	Enables reading to skip ahead to the next file in a sequence upon encountering a corrupt block (Guardian Error 59).

Extract parameters: Reporting and error handling (continued)

Parameter	Description
HOMETERMMESSAGES	Suppresses message display to the home terminal.
LAGSTATS	Specifies parameters for capturing lag and related statistics.
LIST | NOLIST	Control whether or not 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
SUPPRESSALTERMESSAGES | NOSUPPRESSALTERMESSAGES	Suppress the messages produced when Extract finds a FILE ALTER record in the TMF audit trail.
SUPPRESSFILEOPMESSAGES | NOSUPPRESSFILEOPMESSAGES	Suppresses the output of messages generated after FILE RENAME, PURGE, CREATE, ALTER, SETMODE.
SUPPRESSMARKERMESSAGES	Suppresses messages generated when markers are processed.
VERSIONERR	Specifies error handling when database definitions are out of synchronization with audit trail records.

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 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.

Replicat 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 commands from a different parameter file.
SETENV	Sets a NonStop environment variable.

Replicat parameters: Process management

Parameter	Description
ALLOCFILES	Controls the amount of incremental memory that is allocated when the amount of memory specfied 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 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.
DISCARDFILE	Contains records that could not be processed.
DISCARDROLLOVER	Specifies times at which to create new discard files.

Replicat parameters: Process management (continued)

Parameter	Description
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.
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 GoldenGate trails.
FILEAGEDAYS	Specifies the number of days a file can be inactive before it is aged off Replicat's file list.
FILEOPWARNING	Controls Replicat's behavior when it attempts to purge non-existent files.
FLUSHCHECKPOINT | NOFLUSHCHECKPOINT	Controls whether or not files are flushed from a 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.
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.

Replicat parameters: Process management (continued)

Parameter	Description
NETWORKCHECKPOINTS	When Replicat encounters partitions residing on a remote node, triggers update of the local checkpoint file to allow identification of replicated data.
NUMFILES	Controls the initial allocation of memory dedicated to storing information about files to be processed by GoldenGate.
PARTMAP	Specifies alternate partitioning schemes during file creation on the backup system.
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	Purges 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 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 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 GoldenGate trails from run to run. Either REPLICAT or SPECIALRUN is required.
WAITFILEEVENT	Waits for a file-related event to occur before proceeding.

Replicat parameters: Security

Parameter	Description
DECRYPTTRAIL	Use DECRYPTTRAIL to decrypt encrypted GoldenGate trail files.
LOGON	Specifies an alternate user ID under which to run Replicat.

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.
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 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.

Replicat parameters: Data mapping

Parameter	Description
COLMATCH	Sets global column mapping rules.
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.

Replicat parameters: Data formatting and conversion

Parameter	Description
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 or not the Y2K window changes the century.

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 a GoldenGate macro.
MACROCHAR	Changes the macro character to something other than #.

Replicat parameters: Data customization

Parameter	Description
EMBEDDEDMACROS | NOEMBEDDEDMACROS	Processes macros embedded in a quoted string.
TALUSEREXIT	Invokes TAL user exit routines at different points during processing.

Replicat parameters: Reporting and error handling

Parameter	Description
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 or not 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.
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 done.
RETRYDELAY	Specifies the delay between attempts at retrying a failed insert, update, or delete operation.
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.

Replicat parameters: Reporting and error handling

Parameter	Description
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.

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.
READER	Triggers the creation of a Reader process to access a local trail and communicate with the system Coordinator

Syncfile parameters summary

Syncfile allows you to schedule and manage file duplication when you wish to copy files in their entirety.

Syncfile parameters: All

Parameter	Description
ABORTDUPERRWINDOW	Specifies a time period for Syncfile to restart aborted processes before ending abnormally.
CHECKINTERVAL	Allows you to 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.

Syncfile parameters: All (continued)

Parameter	Description
DUPONLYAFTEREVENT	Specifies a time for events to execute.
DUPPROCESS	Allows you to 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.

Alphabetical reference

This begins the alphabetical reference for all GoldenGate parameters on the NonStop platform.

ABORTDUPERRWINDOW

Valid for Syncfile

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>;`

Argument	Description
`<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

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.

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 GoldenGate environment prefix, GGS.
- The default volume and subvolume for parameter files, $VOL.GGSPARM.
- The default volume and subvolume for report files, $VOL.GGSRPT.
- The default volume and subvolume for AUDCFG, $SYSTEM.GGS.AUDCFG.

> **NOTE** When you use a DEFINE, it automatically includes the node of any file path you specify. So, if you are adding defines for remote objects, you must indicate the fully qualified file name, with its node, otherwise the value you specify may not resolve correctly.

Defaults set by ADD DEFINE for =GGS_PARAMS or =GGS_REPORT can be overridden by the appropriate GGSCI ADD or ALTER command.

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>]`

Argument	Description

`<define>`	The <define> must be one of the following:
	`=GGS_PARAMS`
	Changes the default subvolume used to store parameter files.
	`=GGS_REPORT`
	Changes the default subvolume used to store the report file.
	`=GGS_AUDCFG`
	Changes the default file name of the AUDCFG file.
	`=GGS_DB_SUBVOL`
	Defines the subvolume location of the GGS database.

Argument	Description
	=GGS_PREFIX Allows you to define a different GoldenGate prefix for process names and subvolumes.

<class, node/vol spec>	If you are defining new volume and subvolumes, or a file name enter <class, 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>`

Example This example changes the default subvolume used to store parameter files to $GGSPROD.

```
ADD DEFINE =GGS_PARAMS, CLASS DEFAULTS, VOLUME \PROD.$DATA3.GGSPROD
```

Example This example changes the default subvolume used to store the report files.

```
ADD DEFINE =GGS_REPORT, CLASS DEFAULTS, VOLUME \PROD.$DATA3.NEWRPT
```

Example This example changes the location of the audit configuration file.

```
ADD DEFINE =GGS_AUDCFG, CLASS MAP, FILE \NODE.$DATA1.GGS.AUDCFG
```

Example 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 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
```

NOTE For more information on working with defines, see the *GoldenGate for HP NonStop Administrator Guide*.

ALLOCFILES

Valid for Extract, Replicat

Use ALLOCFILES to control the incremental number of memory structures allocated once the initial memory allocation specified by the NUMFILES parameter is reached (see page 232). 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 <number of structures>`

Argument	Description
`<number of 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. GoldenGate's memory structures support up to two million tables.

Example `ALLOCFILES 1000`

ALTFILERESOLVE | NOALTFILERESOLVE

Valid for Extract

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.

```
2006-02-16 08:10:58.161307 Scanning $*.A12GEN* for Alternate Key Files
2006-02-16 08:10:58.230067 Files scanned 16
2006-02-16 08:10:58.230788 Scanning $*.A12SQL* for Alternate Key Files
2006-02-16 08:10:58.289293 Files scanned 72
```

ALTINPUT

Valid for Extract

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 <number of bytes>]
[, JTSOFFSET <byte offset>]
[, TANDEMTSOFFSET <byte offset>]
[, ONEFILE]
```

Argument	Description
`RANGE (x OF y)`	Use RANGE to process multiple files within a sequence simultaneously. For information on specifying the RANGE option see page 122
`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.

Argument	Description
FASTREADS	Causes Extract to perform bulk reads of the source file set, boosting read performance.
WAITNEXTRA <number of 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 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 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

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

AUDITING

Valid for GLOBALS

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}

AUDITREPS | NOAUDITREPS

Valid for Replicat

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

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>
```

Argument	Description
<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

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>

Argument	Description
<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

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>...]

Argument	Description
<cpu num>	The CPU identifier.

AUDSERVPARAM

Valid for Extract

Use AUDSERVPARAM to pass parameters that are specific to the Audserv process.

AUDSERVPARAM has the following dependencies:

- AUDSERVPARAM GETPURGEDATAS is required before a file list in the Extract parameter file.
- Guardian must be version D46 or G06 and later.

Default GETNONDATACHANGES, IGNOREPURGEDATAS, IGNOREALTFILES, SQLCATCLOSEDELAY 60

Syntax

```
AUDSERVPARAM
[ABENDONSECURITYCHECK | NOABENDONSECURITYCHECK]
[GETNONDATACHANGES | NOGETNONDATACHANGES]
[GETPURGEDATAS | IGNOREPURGEDATAS]
[GETALTFILES | IGNOREALTFILES]
[EXCLUDEFILECODES (<filecode> [, ...])
[SQLCATCLOSEDELAY <seconds>]
```

Argument	Description
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.
GETNONDATACHANGES | NOGETNONDATACHANGES	Filters records for SQL partition moves and splits. When using NOGETNONDATACHANGES, you cannot extract any data changes produced by the RDF subsystem.
GETPURGEDATAS | IGNOREPURGEDATAS	Use GETPURGEDATAS | IGNOREPURGEDATAS to control whether or not Extract writes TMF audited Enscribe purge data operations to a trail. GoldenGate supports PURGEDATA for both Format1 and Format2 Enscribe files for the base table and all secondary partitions, as well as SQL tables. The default is IGNOREPURGEDATAS. **Note:** GoldenGate supports GETPURGEDATA on partitioned drives with an option called PARTONLY. However, you must apply PARTONLY to the primary partition for this to work. GoldenGate does not support PURGEDATA on a secondary partition.

GETALTFILES | IGNOREALTFILES	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 may use the GETALTFILES option in your Extract parameter file as follows:
	`AUDSERVPARAM GETALTFILES`
	The default setting for this option is IGNOREGETALTFILES

Argument	Description
EXCLUDEFILECODES (<filecode>)	The numeric file code of a type of file whose audit is to be excluded. **Note:** 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

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>
```

Argument	Description
<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 may not use AUDSERVPREFIX, and vice versa.

AUDSERVPROCESS

Valid for Extract

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>
```

Argument	Description
<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 may not use AUDSERVPREFIX, and vice versa.

AUDSERVPROGRAM

Valid for Extract

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 file 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

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>]
```

Option	Description
`<process type>`	Specify one of the following: ◆ EXTRACT or EXT ◆ REPLICAT or REP ◆ ER (Extract and Replicat) ◆ LOGGER ◆ SYNCFILE or SYNC ◆ COORD (Coordinator)
`<group type>`	A group name or wildcard specification for multiple groups. When wildcarding is used, 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

The AUTOSTART parameter specifies processes that are to be automatically started by Manager. When a Manager process starts up it scans the list and attempts to start any processes that are not already running. The list is not reprocessed in response to a REFRESH command.

Syntax

```
AUTOSTART [<group type>] {<group name> | <process name>}
      [, ALLPROCESSES]
```

Option	Description
`<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 may 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 may be used for all or part of the name.
`ALLPROCESSES`	Specifies that TASK groups should be included in wildcarded groups to be started.

Example The following will start all Replicat processes that begin with R20.

```
AUTOSTART REPLICAT R20*
```

Example The following will start all Extracts.

```
AUTOSTART EXT *
```

Example The following will start the logger named $ABC01.

```
AUTOSTART LOGGER $ABC01
```

Example The following will start all groups that begin with R20 except TASK groups.

```
AUTOSTART R20*
```

Example The following will start all groups, including TASK groups, that begin with R20.

```
AUTOSTART R20*, ALLPROCESSES
```

BACKUPCPU

Valid for Manager

Use BACKUPCPU to specify a CPU that is running the backup Manager process.

Syntax

```
BACKUPCPU <cpu number>
```

Argument	Description
<cpu number>	The identifier for the CPU that is running the backup Manager process.

BEGIN

Valid for Extract, Replicat

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>]
```

Argument	Description
<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.

Example BEGIN 2006-08-12 08:00 specifies a timestamp for August 12, 2006 at 8:00 AM.

Example BEGIN 2006-08-12 08:00:30.000030 specifies a timestamp for August 12, 2006 at 8:00:30.000030 AM.

BULKIOLOAD | NOBULKIOLOAD

Valid for Replicat

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.

Syntax

```
BULKIOLOAD | NOBULKIOLOAD
```

CHECKINTERVAL

Valid for Syncfile

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>;
```

Argument	Description
`<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

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>
```

Argument	Description
`<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

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

CHECKPOINTSECS

Valid for Extract and Replicat

Use the CHECKPOINTSECS parmeter 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 GoldenGate Technical Support.

Default 10

Syntax CHECKPOINTSECS <seconds>

Argument	Description
<seconds>	The number of seconds to wait before issuing a checkpoint.

Example CHECKPOINTSECS 20

CHECKUNIQUEKEY | NOCHECKUIQUEKEY

Valid for Replicat

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

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>

Argument	Description
<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

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

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 may also use the COLMAP option of the Replicat MAP parameter.

Syntax
```
COLMATCH
{NAMES <target column> = <source column> |
PREFIX <prefix> | SUFFIX <suffix> | RESET}
```

Argument	Description
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.

Argument	Description
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	**ORD Table**	**ACCOUNT Table**	**ORDER Table**
CUST_CODE CUST_NAME CUST_ADDR PHONE	CUST_CODE CUST_NAME ORDER_ID ORDER_AMT	CUSTOMER_CODE CUSTOMER_NAME CUSTOMER_ADDRESS PHONE	CUSTOMER_CODE CUSTOMER_NAME ORDER_ID 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_NAME ORDER_ID ORDER_AMT	CUST_CODE_K CUST_NAME_K ORDER_ID 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

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 hypens 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 GoldenGate NSK SQL Extract parameter file
-- GoldenGate NSK SQL Extract parameter file
```

COMPRESSDELETES | NOCOMPRESSDELETES

Valid for Extract

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, 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 for your transactional data management system.

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

COORDINATOR

Valid for Coordinator, Replicat

Coordinator

Use COORDINATOR in the Coordinator parameter file to identify the name of the group.

Syntax
```
COORDINATOR <group name>
```

Argument	Description
<group name>	The name of the group.

Example
```
COORDINATOR ACCTCO
```

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 <centisecs>]
```

Argument	Description
<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 <centisecs>	Sets the number of seconds or centiseconds that Replicat will wait 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
```

CPU

Valid for Logger

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>
```

Argument	Description
<primary>	The primary CPU identifier.
<backup>	The backup CPU identifier.

CUSEREXIT

Valid for Extract, Replicat

Use CUSEREXIT to call custom C routines at different points during processing. If your user exit is written in COBOL, see page 133 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

Use DEBUGONSTACKCHECK to call DEBUG whenever an application's process stack is close to overflowing. GoldenGate recommends using this parameter only when instructed to do so by GoldenGate support.

Default Omit DEBUGONSTACKCHECK

Syntax

```
DEBUGONSTACKCHECK
```

DECRYPTTRAIL

Valid for Extract, Replicat

Use DECRYPTTRAIL to decrypt GoldenGate trails that were encrypted by an upstream Extract process (for information on ENCRYPTTRAIL see page 146). Specify DECRYPTTRAIL only when the ENCRYPTTRAIL parameter is specified in the upstream Extract process.

Syntax

```
DECRYPTTRAIL
```

DICTIONARY

Valid for Extract, Replicat

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 <dict subvol name>
```

Argument	Description
`<dict subvol name>`	A physical subvolume or an existing define name of class catalog.

DISCARDFILE

Valid for Extract, Replicat

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.

Syntax

```
DISCARDFILE <file name>
[, APPEND | PURGE] | ROLLOVER]
[, EXTENTS (<primary>, <secondary>, <maximum>)]
[, MEGABYTES <megabytes>]
[, OWNER (<group num>, <user num>)]
[, SECURE “<rwep>”]
```

Argument	Description
`<file name>`	A physical file name or an existing define name of class map.
`APPEND | PURGE | ROLLOVER`	◆ APPEND appends records to an existing file. ◆ PURGE purges the file of existing records, then adds records from the current process. ◆ ROLLOVER renames the discard files by appending a sequence number to the file name, according to these rules: If the file name is 6 characters or less, 2 digits are appended. If the file name is 7 characters, 1 digit is appended. If the file name is 8 characters, 1 digit overlays the last character of the file name.

Argument	Description
`EXTENTS (<primary>, <secondary>, <maximum>)`	Sets up the extent sizes and maximum extents for the file.
`MEGABYTES <megabytes>`	Sets the maximum size of the file in megabytes. The maximum size is 2 gigabytes.
`OWNER (<group num>, <user num>)`	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", APPEND`

DISCARDROLLOVER

Valid for Extract, Replicat

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>.<extension>, 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 is required. Both options can be used together. Using AT without ON creates a new discard file at the specified time every day.

Default Disabled. No rules specified.

Syntax

```
DISCARDROLLOVER {
AT <hh:mi> |
ON <day of week> |
AT <hh:mi> ON <day of week>
}
```

Argument	Description
`AT <hh:mi>`	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. ◆ <mi> is minutes from 00 through 59.
`ON <day of week>`	The day of the week to age the file. Valid values: SUNDAY through SATURDAY They are not case-sensitive.

Example `DISCARDROLLOVER AT 05:30`

Example `DISCARDROLLOVER ON friday`

Example `DISCARDROLLOVER AT 05:30 ON FRIDAY`

DISKTHRESHOLD | NODISKTHRESHOLD

Valid for Manager

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 may use the THRESHOLD or NOTHRESHOLD parameters to report on disk or tape audit thresholds. For information on these parameters see page 273.

Default `DISKTHRESHOLD 20%`

Syntax `DISKTHRESHOLD <percent left> | NODISKTHRESHOLD`

Argument	Description
`<percent left>`	The percentage level at which to generate an event message.

DISPLAYTRAILSWITCH | NODISPLAYTRAILSWITCH

Valid for Extract

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

Use DOWNCRITICAL to include a process that has terminated normally 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. Processes that terminate abnormally are automatically reported by the DOWNREPORT parameter.

See the DOWNREPORT parameter page 141 for information on setting the frequency for sending information to the event log.

Default Manager does not report a normally terminated process.

Syntax
```
DOWNCRITICAL
```

DOWNREPORT

Valid for Manager

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>
```

Argument	Description
`<minutes>`	The reporting interval, in minutes.
`<hours>`	The reporting interval, in hours.

DUP

Valid for Syncfile

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 or not 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>];
```

Argument	Description
<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 (<code list>)	A comma delimited list to specify the filecodes 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 filecode that is not in the list will be excluded.
EXCLUDEFILECODE (<code list>)	A comma delimited list to specify the filecodes 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 filecode 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.

Argument	Description
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" on page 143.

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;
```

For more information on this see the *GoldenGate for HP NonStop Administrator Guide*.

DUPONLYAFTEREVENT

Valid for Syncfile

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

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>;`

Argument	Description
`<process name>`	The process name used to duplicate files.

DYNAMICPARTITIONS

Valid for Extract

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.

Syntax `DYNAMICPARTITIONS`

Default Not activated

DYNAMICPORTLIST

Valid for Manager

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>} [, ...]`

Argument	Description
`<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

Use DYNAMICPORTREASSIGNDELAY to specify the time to wait before a port can be reused.

Default 3

Syntax `DYNAMICPORTREASSIGNDELAY <time>`

Argument	Description
`<time>`	The number of seconds to delay before reusing a port.

EMBEDDEDMACROS | NOEMBEDDEDMACROS

Valid for Extract, Replicat

Use EMBEDDEDMACROS to control whether or not 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

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}`

Argument	Description
`<collector>`	Specify either the Collector name, or NONE when there is no Collector.

ENCRYPTTRAIL | NOENCRYPTTRAIL

Valid for Extract

Use ENCRYPTTRAIL to encrypt data records in subsequent GoldenGate trails until a NOENCRYPTTRAIL is encountered. All records going into a 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

Use END to specify the point at which the process stops processing in the TMF audit or 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 2006-08-12 17:00:00.

END is used to determine when SPECIALRUN processing will terminate.

Syntax `END {<date> [<time>] | RUNTIME}`

Argument	Description
`<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

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]
```

Argument	Description
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.

Argument	Description
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 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 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

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 a GoldenGate trail.

Default 1 second

Syntax

```
EOFDELAY <seconds> | EOFDELAYCSECS <centiseconds>
```

Argument	Description
<seconds>	The number of seconds to delay.
<centiseconds>	The number of centiseconds to delay.

ERREPLYTIMEOUT

Valid for GLOBALS

Use ERREPLYTIMEOUT to set the timeout, in seconds, when GGSCI communicates with GoldenGate components.

Default 30

Syntax

```
ERREPLYTIMEOUT <seconds>
```

Argument	Description
<seconds>	Specify a timeout value in seconds.

ERROR59ROLLOVER

Valid for Extract

Use ERROR59ROLLOVER during direct file extraction to tell Extract to skip ahead to the next file in a sequence upon encountering a corrupt block (Guardian Error 59).

Certain applications leave file blocks in a corrupt state to signal that the end of data within the file has been reached. All data prior to the first corrupt block is considered valid. Extract, when processing a sequence of files directly, will correctly process all data up until the corrupt block. By default, if Extract subsequently reads a corrupt block, it will abend.

Syntax

```
ERROR59ROLLOVER
```

ETNEWFORMAT | ETOLDFORMAT

Valid for Extract

Use ETNEWFORMAT to cause Extract to generate trails in formats that are compatible with Replicat version 7.0 or later.

Use ETOLDFORMAT to cause Extract to generate trails in formats that are compatible with versions of Replicat prior to 7.0. You must use this parameter before any other parameter when running pre-7.0 versions of GoldenGate for NonStop, and pre-7.2 versions of GoldenGate for Windows and UNIX.

Default

```
ETNEWFORMAT
```

Syntax

```
ETNEWFORMAT | ETOLDFORMAT
```

EVENT

Valid for Syncfile

Use EVENT to define an event and its schedule. Associate the event with a file set via 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>];
```

Argument	Description
`<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

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>]
```

Argument	Description
`<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.*).

Example

```
EXCLUDEFILE $DATA4.DAT.TRANSFL
```

Example

```
EXCLUDEFILE $D15.DAT.*
```

EXCLUDEGGSTRANSRECS

Valid for Extract

Use EXCLUDEGGSTRANSRECS to suppress the creation of network transaction tracking records. This parameter should not be used if you have configured Readers and a Coordinator process for your system, because these processes depend on tracking records in the trail. Use this parameter only if you are not replicating distributed network transactions.

Default Create network transactions tracking records

Syntax

```
EXCLUDEGGSTRANSRECS
```

EXCLUDESUFFIXCHAR

Valid for Manager

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 DEFINES for GGS_PREFIX (see page 118), 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 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>

Option	Description
<characters>	The characters to be excluded. These should be entered as a string of characters without commas. The characters may optionally be enclosed in single or double quotation marks (e.g. "WTZ" or 'BP').

Example The following examples all exclude the characteres T and X from the first position of the suffix for generated process names.

```
EXCLUDESUFFIXCHAR TX
EXCLUDESUFFIXCHAR 'TX'
EXCLUDESUFFIXCHAR "TX"
```

Example To reset to the defaults, enter an empty set of characters as shown below.

```
EXCLUDESUFFIXCHAR ""
```

EXPANDDDL

Valid for Extract, Replicat

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 FORMATASCII or FORMATSQL are specified.

Syntax

```
EXPANDDDL <format>
[, ZEROFILL <width> | ARRAYWIDTH]
[, INCLUDEREDEFS | OMITREDEFS]
```

Argument	Description
`<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.

EXTFILE

Valid for Extract, Replicat

Use EXTFILE when Extract writes to a GoldenGate trail that contains a single file. EXTFILE defines the name, dimensions, and security of a file in the 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 <megabytes>]
[, OWNER (<group num>, <user num>)]
[, SECURE “<rwep>”]
```

Argument	Description
`<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 <megabytes>`	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 num>, <user num>)`	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

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>`

Argument	Description
`<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

Use EXTTRAIL to establish the current 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 a 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 GoldenGate trails, refer to page 56.

Syntax `EXTTRAIL <file prefix>`

Argument	Description
`<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

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

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 GoldenGate trails, SOURCEISFILE, or direct file reads.

Default FASTPOSITION

Syntax FASTPOSITION

FASTREADS | NOFASTREADS

Valid for Coordinator, Extract, Replicat

Use FASTREADS to change the number of bytes that Extract reads when processing 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 a GoldenGate trail and does not apply when reading TMF audit trails, since GoldenGate always reads audit trails with a large block read.

Default NOFASTREADS

Syntax FASTREADS

FETCHCOMPS | FETCHLASTIMAGE

Valid for Extract

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

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>];
```

Argument	Description
`<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 page 158.

Argument	Description
<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" on page 158 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 may use other parameters as options with FILE; this allows you to 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>]
;
```

Filter	Description
`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};
```

Attribute	Description
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];
```

Attribute	Description
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.

Attribute	Description
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 may 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>;
```

Attribute	Description
<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};
```

Attribute	Description
SUSPENDED	Temporarily suspends logging for the particular file set.
ACTIVE	Resumes logging for the file set.

FILE | TABLE

Valid for Extract

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 lease one FILE or TABLE statement per parameter file is required.

Syntax

```
FILE <file name>
[, AUTOTRUNCATE]
[, COMPRESSDELETES]
[, DEF <source ddl definition name>]
[, EXITMAP]
[, EXITPARAM "<exitparam string>"]
[, SQLNAME <table alias>]
[, TARGETNAME <target file name>]
[, PARTITIONS <partition spec>]
[, WHERE (<where condition>)]
[, FILTER (<expression>)]
[, RANGE (<range specification>)]
[, USETARGETDEFLENGTH]
[, KEYCOLS (<key column specification>)]
[, COLMAP (<column mapping specification>) | NOCOLMAP]
[, SQLEXEC (<SQLEXEC clause>)]
[, TARGET <target file name>]
[, TARGETDEF <target ddl definition name>]
[, TOKENS (<token specification>)]
[, ALTNAME <alternate file name>]
;
```

Argument	Description
`<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

Option	Operation
`COMPRESSDELETES`	See “Compressing records” on page 163.

Option	Operation
DEF FILTER PARTITIONS RANGE WHERE	See "Selecting records" on page 163.
COLMAP EXITMAP SQLNAME TARGETDEF TARGETNAME USETARGETDEFLENGTH	See "Mapping data" on page 166.
AUTOTRUNCATE	See "Purging records for an initial load" on page 168.
ALTNAME	See "Handling missing files" on page 168.
EXITPARAMS	See "Passing literal strings to user exits" on page 169.
SQLNAME <table alias>	See "Specifying a table alias" on page 169.
SQLEXEC	See "Performing a query" on page 169.
TOKENS	See "Using tokens" on page 172.

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, 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.

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>]
;
```

Argument	Description
`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>];
```

Argument	Description
`<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 on page 164.
`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 8 Permissible WHERE operators

Operator	Example
Column names	`PRODUCT_AMT`

Table 8 Permissible WHERE operators

Operator	Example
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 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);
```

Argument	Description
`(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 page 308 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 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 spec>);
```

Argument	Description

`<volume spec>`	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);`

Mapping data

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 the source and target record lengths” on page 167.

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 mapping specification>) | NOCOLMAP
TARGET <target file name>
[, DEF <source ddl definition> | TARGETDEF <target ddl definition name>]
;
```

Argument	Description
`<column mapping 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 a 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 name>`	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 the 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>;
```

Argument	Description
`<target file name>`	The name of the target file for which Replicat retrieves the layout.

Mapping with a user exit

Use EXITMAP when relying solely on a user exit for column mapping. Omit COLMAP when mapping with user exits.

Syntax

```
FILE <file name>, TARGET <target name>, EXITMAP;
```

Purging records for an 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 IGNORETRUNCATES 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 a GoldenGate or Logger trail.

Syntax
```
FILE <file name>, ALTNAME <alternate file name>
[, DEF <source ddl definition name>];
```

Argument	Description
`<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 or FORMATSQL 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 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.

Important! 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>]
[, ...]
)
```

Argument	Description
`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.

Argument	Description

	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 or not 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. 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 a 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.

Argument	Description
	◆ 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. 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 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> [, ...])
;
```

Component	Description
<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 a 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

Use FILEAGEDAYS to specify the number of days a file can be inactive before the Extract or Replicat process ages it off of its file list.

Only files added with a wildcard match are eligible to be removed. Files cannot be aged off if they are designated in FILE and MAP statements that have filters, SQLEXEC statements, or column mappings that use column functions.

Audserv will inherit this parameter from the Extract.

Default Files are not aged off the list.

Syntax

```
FILEAGEDAYS <days>
```

Argument	Description
<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.

Example The following MAP example includes the @DATE function in the column mapping so the TL* files are not eligible for aging off the file list.

```
MAP $DATA.SOURCE.TL*, TARGET $DATA4.TARGET.*
COLMAP (USEDEFAULTS,
       DATETIMECOL = @DATE(“YYYY-MM-DD:HH:MI:SS”, “YYYY-MM-DD:HH:MI:SS”,
       @DATENOW())
       );
```

Example The following MAP example does not include a function, so the TL* files are eligible to be aged off the file list.

```
MAP $DATA.SOURCE.TL*, TARGET $DATA4.TARGET.*
COLMAP (USEDEFAULTS,
       DATETIMECOL = “2009-02-14:01:02:03”
       );
```

FILEEXCLUDE

Valid for Extract

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 ID>`

Argument	Description
`<file ID>`	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

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}`

Argument	Description
ABEND	Stops processing
IGNORE	Ignores the purge attempt and continues processing.
WARN	Issues a warning to the report file.

FILERESOLVE

Valid for Extract

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 wildcarded file lists, 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}

Argument	Description
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.

Example 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 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 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

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 is automatically turned on when extracting data directly from files or tables (SOURCEISFILE parameter). Turn this functionality off with NOFILLSHORTRECS.

Also see the USESOURCERECLENGTH option of the FILE parameter.

Default NOFILLSHORTRECS

Syntax FILLSHORTRECS | NOFILLSHORTRECS

FILTERVIEW | NOFILTERVIEW

Valid for Extract

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

Use FLUSHCHECKPOINT to control whether or not 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

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>`

Argument	Description
`<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

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>`

Argument	Description
`<centiseconds>`	The delay interval in centiseconds.

FORCEUSESYSKEY | NOFORCEUSESYSKEY

Valid for Replicat

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 assuance 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

Example 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 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

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 newline 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 newline character.

After the last record in each transaction is output, a commit record will appear. This record contains C, the delimiter and a newline 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	Description
BCP	Formats the output for compatibility with SQL Server's BCP (Bulk Copy Program) high-speed load utility.
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.
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>	An alternative field delimiter (the default is tab). Use the word TAB to delimit with tabs, otherwise use a single character enclosed in single quotes (for example, '/').
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).

Option	Description
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 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,2006-02-17:14:09:46.421335,8,1873474,
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|2006-02-17:14:09:46.421335|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,2006-02-17:14:09:46.421335,Eric,San Fran,550,
V,CUSTOMER,2006-02-17:14:09:46.421335,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.

FORMATLOAD

Valid for Extract

Use FORMATLOAD to output records to the EXTFILE or RMTFILE that conform to the format expected by SQLCI LOAD or COPY. When you specify FORMATLOAD, header informatio is not included in the output so that data in the GoldenGate file is in the correct format to load into a target file or table. As a result, you can specify only one TABLE or FILE entry per EXTFILE and cannot use EXTTRAIL entries.

FORMATLOAD discards compressed update records (as if IGNORECOMPS were set). It includes full image updates and deletes unless you explicitly exclude them with the IGNOREUPDATES and IGNOREDELETES parameters.

When specifying FORMATLOAD and performing custom mapping in a user exit, you must specify a TARGET file name for the corresponding FILE or TABLE entry. The target file must reflect the structure of the record after mapping so that it is formatted correctly for the load utilities.

To load a GoldenGate file with this format into a SQL table, use SQLCI in the following manner.

```
>SQLCI
>>LOAD <extract file>, <sql table>, RECIN <load rec length>;
```

In the preceding example, <load rec length> is the length reported in the GoldenGate file (this may be different than the record length reported by FUP). This record length can also be obtained from the SQL catalog with the following query.

```
>>SELECT RECORDSIZE FROM FILES WHERE FILENAME = "<sql table name>";
```

Syntax FORMATLOAD

FORMATLOCAL

Valid for Extract

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

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 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 newline character

After the last record in each transaction is output, a commit record will appear. This record contains C, the delimiter and a newline 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}`

Option	Description
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('2007-05-01','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.

FUNCTIONSTACKSIZE

Valid for Extract, Replicat

Use FUNCTIONSTACKSIZE to control the size of the memory stack that is used for processing GoldenGate functions. You should not need to use this parameter unless GoldenGate returns a message indicating that the size of the stack should be increased.

The memory stack holds arguments supplied to and from a 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 GoldenGate's performance and its usage of system memory. Increasing this parameter can adversely affect GoldenGate's performance and use of system memory.

FUNCTIONSTACKSIZE must appear in the parameter file before any parameter clauses are listed.

Default 200

Syntax `FUNCTIONSTACKSIZE <stacksize>`

Argument	Description
<stacksize>	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

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

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 page 190 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

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>

Argument	Description
<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

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

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

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

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

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 page 297 for more information on the GETENV variables.

Default None

Syntax
```
GETENV (<environment variable>)
```

Argument	Description
`<environment variable>`	The name of the environment variable

Example
```
GETENV TRAIL1
```

GETFILEOPS | IGNOREFILEOPS

Valid for Extract, Replicat

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

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

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

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
```

GETNEWCOLUMNS | IGNORENEWCOLUMNS

Valid for Extract, Replicat

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 allows you to replicate columns added to a table after GoldenGate startup.
- Downstream in Replicat (NonStop only), replicates the statement that created the column.

Apply GETNEWCOLUMNS to files or 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 GoldenGate trail.

Default GETNEWCOLUMNS

Syntax GETNEWCOLUMNS | IGNORENEWCOLUMNS

GETPARTONLYPURGEDATAS | IGNOREPARTONLYPURGEDATAS

Valid for Extract, Replicat

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.

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

Includes or excludes file purge operations.

Default IGNOREPURGES

Syntax GETPURGES | IGNOREPURGES

GETPURGEDATAS | IGNOREPURGEDATAS

Valid for Extract, Replicat

Use GETPURGEDATAS or IGNOREPURGEDATAS to control whether or not Extract writes purge data operations to a trail. GoldenGate supports PURGEDATA for both Format1 and Format2 Enscribe file base tables and all secondary partitions, as well as SQL tables.

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

Includes or excludes file rename records.

Default IGNORENAMES is the default unless GETFILEOPS is used.

Syntax GETRENAMES | IGNORERENAMES

GETREPLICATES | IGNOREREPLICATES

Valid for Extract

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 page 184 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

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> 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 GoldenGate with the FASTUNLOADSHARED parameter.

Default IGNOREROLLBACKS

Syntax GETROLLBACKS | IGNOREROLLBACKS

GETUPDATEAFTERS | IGNOREUPDATEAFTERS

Valid for Extract, Replicat

Use GETUPDATEAFTERS or IGNOREUPDATEAFTERS to include or exclude after images of update records from a specified file in a 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

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

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

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>

Argument	Description
`<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

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 may 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

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>

Argument	Description
<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

Use HOMETERM to change the default GoldenGate system home terminal to another terminal.

Default The terminal residing on the system on which GoldenGate is installed.

Syntax HOMETERM <home terminal name>

Argument	Description
`<home terminal name>`	Specify a terminal, such as $VHS.

HOMETERMMESSAGES

Valid for GLOBALS

Use HOMETERMMESSAGES to suppress or direct GoldenGate messages to the home terminal. By default, GoldenGate processes write messages to both the home terminal and EMS.

This parameter can be overridden in other parameter files by specifying a value for HOMETERMMESSAGES that is different from the GLOBALS value.

Default YES

Syntax
```
HOMETERMMESSAGES {YES | NO}
```

Argument	Description
`YES | NO`	The default is YES, write to the home terminal and EMS. Specify NO to suppress writing to the home terminal and to write only to EMS.

HOST

Valid for GLOBALS

The HOST parameter does the following:

- Identifies the remote NonStop system to communicate with GGSCI.
- Helps 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 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 <subvol>]
[, NODENUM <node number>]
```

Argument	Description
`<system name>`	Identifies the host system name.
`GGSSUBVOL <subvol>`	Identifies the subvolume. Required with NODENUM.

Argument	Description
NODENUM <node number>	Identifies the NonStop node.

Example The first example below identifies a 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 This example identifies multiple nodes that contain data and a 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

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 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

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

Use INCLUDE to include a macro library in a parameter file.

Default None (no file included)

Syntax `INCLUDE <file name>`

Argument	Description
<file name>	The full path to the library file.

Example The following example includes macro library DATELIB.

```
INCLUDE $DATA4.GGSMACR.DATELIB
```

INITAUXPOSITION | NOINITAUXPOSITION

Valid for Extract

Use NOINITAUXPOSITION to override the default Audserv position in an AUX trail. The default is to position Audserv into an AUX trail using the contents of the first AUXPTR record for each AUX trail that is found in the MAT. Specifying NOINITAUXPOSITION overrides the default and generates a message similar to:

```
Position AUX00: MAT seqno 27, RBA 33437504, AUX seqno 187, RBA, 14734072
TS 2006-01-13 09:51:26.427214
```

Default INITAUXPOSITION

Syntax INITAUXPOSITION | NOINITAUXPOSITION

INSERTALLRECORDS | NOINSERTALLRECORDS

Valid for Replicat

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, however, increases the size of your target tables. As such, 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 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 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

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 INSERTDELETES | NOINSERTDELETES

INSERTMISSINGUPDATES | NOINSERTMISSINGUPDATES

Valid for Replicat

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

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

Under NonStop parallel TCP/IP architecture, a process can have multiple interfaces available. Use IPINTERFACE to restrict the Manager process to the interface specified by an input IP address or DNS name.

If the address specified by the user is not associated with the TCP/IP process, STATUS MANAGER will return the last error.

```
GGSCI> STATUS MANAGER
Manager process $ACMGR is running (IP socket not open error 4115 (Can't
assign requested address, completing bind)).
```

Default Manager will default to all available interfaces using TCP/IP address INADDR_ANY.

Syntax IPINTERFACE {<ip address> | <dns name>}

Argument	Description
<ip address>	The IP address to which Manager is to be restricted.
<dns name>	The domain name to which Manager is to be restricted.

Example The interface restriction may be specified either by using the IPINTERFACE parameter or by attaching the IP address to the TCPIPPROCESSNAME as shown in the second example below.

```
TCPIPPROCESSNAME $ZTC1
IPINTERFACE 10.10.10.20

TCPIPPROCESSNAME $ZTC1@10.10.10.20
```

Example The entry of the address 0.0.0.0 as shown in the following example requests the default behavior of using INADDR_ANY.

```
IPINTERFACE 0.0.0.0
```

LAGCRITICAL

Valid for Manager

Use 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 and LAGCRITICALHOURS 2 specify reporting intervals of 20 seconds and two hours.

When using a LAGCRITICAL parameter, specify LAGREPORT to determine how often the lag times are evaluated and reported.

Syntax

```
LAGCRITICALSECONDS <seconds> |
LAGCRITICALMINUTES <minutes> |
LAGCRITICALHOURS <hours>
```

Argument	Description
<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.

LAGINFO

Valid for Manager

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 2 and LAGINFOHOURS 2 specify reporting intervals of two minutes and two hours.

When using a LAGINFO parameter, specify LAGREPORT to determine how often the lag times are evaluated and reported.

Syntax
```
LAGINFOSECONDS <seconds> |
LAGINFOMINUTES <minutes> |
LAGINFOHOURS <hours>
```

Argument	Description
<seconds>	The reporting interval, in seconds.
<minutes>	The reporting interval, in minutes.
<hours>	The reporting interval, in hours.

LAGREPORT

Valid for Manager

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.

Syntax
```
LAGREPORTMINUTES <minutes> | LAGREPORTHOURS <hours>
```

Argument	Description
<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

Use LAGSTATS to periodically collect, and optionally report, lag and other performance-related statistics to the report file.

Syntax
```
LAGSTATS
[, INTERVAL<unit> <num units>]
[, THRESHOLD<unit> <num units>]
[, REPORT]
```

Argument	Description
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.

Argument	Description
	<unit> may 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

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 <recs>`

Argument	Description
`<recs>`	The number of records to extract.

LIST | NOLIST

Valid for Extract, Replicat

Use LIST and NOLIST to control whether or not 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

Use LOG to identify both a Logger process and the volume (a *log trail*) to which the Logger process writes data. The trails hold data for Replicat and are maintained by 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>
[, MEGABYTES <megabytes> |
   EXTENTS (<primary, secondary, maximum>) |
   NUMFILES <number>]
[, PROCESS <process name>]
[, OWNER <group num>, <user num>]
[, SECURE "<rwep>"]
```

Argument	Description
`<logtrail name>`	The location of the log trail files for the current Logger process and the two-character process prefix. You can use the 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 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” on page 204 for more information.
`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.

Argument	Description
`EXTENTS <primary, secondary, maximum>`	The storage dimensions of each log trail file, where <primary, secondary, max> represents the extents. You must include EXTENTS if you omit MEGABYTES.
`NUMFILES <number>`	Specifies the number of files to include in the log trail.
`OWNER <group num>, <user num>`	The Guardian group and user that owns the log trail files. The default is the group user running GGSCI.
`PROCESS <process name>`	Allows you to 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" on page 204.
`SECURE "<rwep>"`	The Guardian security applied to each of the files. Defaults to the default security of the owner.

Example 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 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, 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 GoldenGate default, enter $vol.GLOGGGL, where $vol is the volume where 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

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>
```

Argument	Description
`<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

Use LOGFILEOPENS to determine how many opens Logger keeps on the current log files.

Default 1

NOTE The default setting is recommended unless otherwise specified by GoldenGate Support.

Syntax
```
LOGFILEOPENS <num opens>
```

Argument	Description
`<num opens>`	The number of opens. Maximum is 8.

LOGGERFILENUM

Valid for Logger

Use LOGGERFILENUM when you are running applications with hard coded file numbers.

As BASELIB opens Logger, Logger file numbers can collide with the hard coded 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 hard coded 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>`

Argument	Description
`<file number>`	The file number on which BASELIB bases its Logger file numbering sequence.

Example `LOGGERFILENUM 30`

LOGGERFLUSHRECS

Valid for Logger, GLOBALS

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 <records>`

Argument	Description
`<records>`	The maximum number of records to buffer before a flush occurs.

LOGGERFLUSHSECS | LOGGERFLUSHCSECS

Valid for Logger, GLOBALS

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 .75 seconds

Syntax `LOGGERFLUSHSECS <seconds> | LOGGERFLUSHCSECS <centiseconds>`

Argument	Description
`<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

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>
```

Argument	Description
<seconds>	The number of seconds for GGSLIB to wait for a response from Logger.

LOGGGSCICOMMANDS

Valid for GLOBALS

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

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>]
```

Argument	Description
`<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 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 `LOGON “super.super”, PASSWORD “ggs123”`

Example `LOGON super.super, PASSWORD & AACAAAAAAAAAAAIALCKDZIRHOJBHOJUH, ENCRYPTKEY superx128`

Example `LOGON super.super, PASSWORD & AACAAAAAAAAAAAIALCKDZIRHOJBHOJUH, ENCRYPTKEY default`

MACRO

Valid for Extract, Replicat

Use MACRO to create a GoldenGate macro. For instructions on creating and using GoldenGate macros, see the *GoldenGate for HP NonStop Administrator Guide*.

The following syntax must be in the order shown.

Syntax
```
MACRO# <macro name>
PARAMS (<p1>, <p2> ,...)
BEGIN
<macro body>
END;
```

Argument	Description
`<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.

Argument	Description
PARAMS (<p1>, <p2>, ...)	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, 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.

Example 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 The following example defines a macro that does not require parameters.

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

Example 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

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>
```

Argument	Description
<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

Use MANAGERREQUIRED to specify whether or not Extract or Replicat can run without a Manager process.

Default NO

Syntax
```
MANAGERREQUIRED {YES | NO}
```

MAP

Valid for Replicat

Use MAP to deliver records from a source to the target. Normally, the source is a 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 name>]
[, DICTIONARY <source ddl dictionary>]
[, EXITPARAM "<exitparam string>"]
[, WHERE (<where condition>)]
[, FILTER (<expression>)]
[, RANGE (<range specification>)]
[, COLMAP ([USEDEFAULTS], <column mapping specification>)]
[, COMPENSCRIBEMAPS | NOCOMPENSCRIBEMAPS]
[, INSERTALLRECORDS]
[, MAPID]
[, PARTIALCOLSOK]
[, SQLNAME]
[, USESOURCERECLENGTH]
[, TARGETDEF <target ddl definition name>]
[, TARGETDICT <target ddl dictionary>]
[, KEYCOLS (<column list>)]
[, USEALTKEY (<key specifier>)]
[, UNMAPPEDALTFILECREATES {ALTFILEVOL | PRIMARYVOL}]
[, CREATETEMPLATE <file>]
[, ALTFILECHAR <num chars>]
[, HANDLECOLLISIONS | NOHANDLECOLLISIONS]
[, DETECTLOCKS]
[, REPERROR (<error>, <response>)]
[, DEFERREDRESOLUTION]
[, PARTMAP <source partition spec>, <target partition spec>]
[, MAPEXCEPTION (<exception specification>)]
[, SHOWSYNTAX]
[, SQLEXEC (<SQLEXEC clause>)];
```

Argument	Description
`<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 name>`	The name of the target file or table.

`DEF`
`DICTIONARY`
`FILTER`
`RANGE`
`WHERE`
	See "Selecting records" on page 212.

Argument	Description
COLMAP COMPENSCRIBEMAPS DEF DICTIONARY INSERTALLRECORDS MAPID PARTIALCOLSOK SQLNAME TARGETDEF TARGETDICT USESOURCERECLENGTH	See “Mapping data” on page 215.
KEYCOLS	See “Defining primary key columns” on page 218.
EXITPARAMS	See “Passing literal strings to user exits” on page 219.
CREATETEMPLATE ALTFILECHAR	See “Creating a target Enscribe file” on page 219.
USEALTKEY	See “Specifying alternate keys” on page 220.
UNMAPPEDALTFILECREATES	See “Replicating file create operations for alternate key files” on page 220
HANDLECOLLISIONS NOHANDLECOLLISIONS	See “Turning error handling on and off” on page 221
DETECTLOCKS	See “Locking records” on page 221.
REPERROR	See “Using REPERROR” on page 222.
EXCEPTIONSONLY MAPEXCEPTION	See “Creating an exceptions statement” on page 223.
DEFERREDRESOLUTION	See “Delaying resolution of file names” on page 223.
PARTMAP	Use PARTMAP to specify alternative mapping of partitions during file creation operations. For details see the PARTMAP parameter on page 234
SHOWSYNTAX MAPID	See “Displaying a SQL statement” on page 224.
SQLEXEC	See “Performing a query” on page 224.

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 may 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>]
)
[, DEF <source ddl definition>]
[, DICTIONARY <source ddl dictionary>]
;
```

Argument	Description
`<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>`	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> is outside the range of error numbers that is used by the database or 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>]
;
```

Argument	Description
`<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 .
`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 9 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 GoldenGate.
Comparison operators	`=, <>, >, <, >=, <=`

Table 9 Permissible WHERE operators

Operator	Example
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);
```

Argument	Description
`(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

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 the source and target record lengths" on page 167.

Syntax

```
MAP <source file name>, TARGET <target file name>,
COLMAP ([USEDEFAULTS], <column mapping specification>)
[, <option>];
```

Argument	Description

`<column mapping specification>`	The column mapping expression, as in
	(<target column> = <source expression>)
	
	Explicitly defines a source-target column map.
	
	<target column> is the name of the target column.
	
	<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 a 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.

Argument	Description

	◆ 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 name>
	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 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 the 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 not produced. Use PARTIALCOLSOK to compress an update fragment with available partial data.

Default Do not compress fragments

Syntax
```
MAP <source file name>, TARGET <target file name>,
DEF <sourcedef>, TARGETDEF <targetdef>, PARTIALCOLSOK;
```

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 page 197 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 GoldenGate’s purposes. 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>] [, ...]);
```

Argument	Description
(<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, altkeys 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, altkey 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.

Syntax

```
MAP <source file name>, TARGET <target file name>,
CREATETEMPLATE <file>, 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>"
```

Argument	Description
"<key specifier>"	The unique key identifier.

Example
```
USEALTKEY "TS"
```

Example 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 may 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 As a best practice, you should provide a MAP statement for both your primary file and alternate key files. 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}
```

Argument	Description
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, GoldenGate waits for DP2 to release its locks before continuing. With the DETECTLOCKS parameter active, 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 page 242). 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>, <response>) [, REPERROR (<error>, <response>)] [, ...];
```

Refer to the REPERROR parameter on page 242 for details on the <error> and <response> specifications. Note that RESET is not a valid option for REPERROR used under MAP.

Example 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 error1 occurs for the first MAP statement, the action should be response2, not response1, because an override was specified. However, if an error1 occurs for the second MAP statement, the response should be response1, the global response. The response for error2 would be response3, which is MAP-specific.

```
REPLICAT <group>
REPERROR (<error1>, <response1>)
MAP <src1>, TARGET <tgt1>, REPERROR (<error1>, <response2>);
MAP <src2>, TARGET <tgt2>, REPERROR (<error2>, <response3>);
```

Example 2 In the following example, when replicating from src1 to src2, all errors and actions (1-3) should apply, because all REPERROR statements address different errors (there are no MAP-specific overrides).

```
REPLICAT <name>
REPERROR (<error1>, <response1>)
MAP <src1>, TARGET <tgt1>, REPERROR (<error2>, <response2>),
REPERROR (<error3>, <response3>);
```

Example 3 In the following example, if error1 occurs for the first MAP statement, the action should be response2. For the second one it would be response1 (the global response), and for the third one it would be 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 <name>
REPERROR (<error1>, <response1>)
MAP <src1>, TARGET <tgt1>, REPERROR (<error1>, <response2>);
MAP <src2>, TARGET <tgt2>, REPERROR (<error2>, <response3>);
REPERROR (<error1>, <response4>)
MAP <src2>, TARGET <tgt2>, REPERROR (<error3>, <response3>);
```

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 may include wildcards and all exceptions that apply to the file set will be written to the <exception file name>. Any valid mapping arguments, such as COLMAP or KEYCOL, may be included.

Syntax

```
MAP <source file name>, TARGET <target file name>,
MAPEXCEPTION (TARGET <exception file name>,
<mapping arguments>);
```

Argument	Description
`<exception file name>`	Identifies the target file for errors that have been identified as EXCEPTION with REPERROR.
`<mapping argument>`	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”)
   )
);
```

Delaying resolution of file names

Replicat resolves wildcarded file names dynamically and fully qualified file names on start up. The latter may not be desirable because a file that is not available when the Replicat

first starts may be available later. This can occur, for example, if the file is in the process of being renamed. DEFERREDRESOLUTION suppresses resolution at start up. Replicat instead retrieves the file attributes when a record for the file is being processed.

Syntax
```
MAP <source file name>, TARGET <target file name>, DEFERREDRESOLUTION;
```

Example This example defers the resolution of $DATA02.ACCT.TAB1 until a record is being processed.

```
MAP $DATA02.ACCT.TAB1, TARGET $DATA05.ACCT.TAB1,
DEFERREDRESOLUTION;
```

The next example sets DEFERREDRESOLUTION for tables on $DATA02.ACCT with names beginning with TAB. Since the default for wildcarded file names is dynamic resolution, processing is not altered in this case.

```
MAP $DATA02.ACCT.TAB*, TARGET $DATA05.ACCT.*,
DEFERREDRESOLUTION;
```

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 page 250.

Performing a query

Use SQLEXEC to perform a SQL Query when processing a record for a SQL/MP table. SQLEXEC enables 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.

Important! 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>] [, ...]
)
```

Argument	Description
`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.”

Argument	Description
	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 or not 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. 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 a 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.

Argument	Description
	◆ 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. 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.

MAPEXCLUDE

Valid for Replicat

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>
```

Argument	Description
<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

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>
```

Argument	Description
<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

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 <recs>`

Argument	Description
`<recs>`	The number of records to discard.

MAXETCHECKPOINTSECS

Valid for Replicat

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 GoldenGate Technical Support.

Default 10

Syntax `MAXETCHECKPOINTSECS <seconds>`

Argument	Description
`<seconds>`	The number of seconds.

MAXTRANSMEM

Valid for Extract

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 <bytes>`

Argument	Description
`<megabytes>`	The number of megabytes. The minimum value is 2000000; the maximum value is 100000000.

Example `MAXTRANSMEM 5000000`

MAXTRANSOPS

Valid for Replicat

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>`

Argument	Description
`<op count>`	The maximum number of operations for a single transaction.

MAXWILDCARDENTRIES

Valid for Extract, Replicat

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>`

Argument	Description
`<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

Use NETWORKCHECKPOINTS to trigger 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. If NETWORKCHECKPOINTS is set, 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 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.

Default Checkpoint files on the partition's remote node are not updated.

Syntax
```
NETWORKCHECKPOINTS
[, MAXRETRIES <num>}
[, DELAYSECS <num> | DELAYCSECS <num>]
```

Argument	Description
`MAXRETRIES <num>`	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 <num>`	Sets the number of seconds that Replicat will wait between tries. The default is 5 seconds and the maximum is 60 seconds.
`DELAYCSECS <num>`	Sets the number of centiseconds that Replicat will wait between tries.

NOTSTOPPABLE

Valid for Logger

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

Use NUMEXTRACTS to override the default number of 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>`

Argument	Description
`<number>`	The number of GoldenGate trails.

NUMFILES

Valid for Extract, Replicat

Use NUMFILES to control the initial number of memory structures allocated to contain information about souce 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 page page 119).

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 <number of structures>`

Argument	Description
`<number of structures>`	The number of memory structures to be allocated. Do not set NUMFILES to an arbitrarily high number, or memory will be consumed unnecessarily. GoldenGate's memory supports up to two million tables.

Example `NUMFILES 4000`

OBEY

Valid for Extract, Replicat

Use OBEY to retrieve parameters from a file other than the current parameter file, then return processing to current parameter file. OBEY commands cannot be nested.

Syntax `OBEY <file name>`

Argument	Description
`<file name>`	The name of the parameter file to obey.

OLDGROUPNAMING

Valid for GLOBALS

Use OLDGROUPNAMING to specify whether or not 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

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 Replicat

Use OPENTIMEOUT or OPENTIMEOUTMINUTES to control when individual Enscribe files are closed after a period of inactivity.

Target Enscribe files are opened when the first replicated database change is received and then remain open to receive additional records. If there is no activity the file will close when the OPENTIMEOUT value is reached. Low values can have a negative effect on performance.

Default 60

Syntax OPENTIMEOUTMINUTES <minutes>

Argument	Description
<minutes>	The inactivity threshold in minutes.

OPENWARNINGS

Valid for Replicat

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 ABEND

Syntax OPENWARNINGS {WARN | IGNORE | ABEND}

Argument	Description
WARN	Issues a warning but continues.

Argument	Description
IGNORE	Continues without warning.
ABEND	Quits processing with a fatal error message.

OVERRIDEDUPS | NOOVERRIDEDUPS

Valid for Replicat

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

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" on page 210.

Syntax PARTMAP <source partition spec>, <target partition spec>

Argument	Description
`<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

Use PARAMCHECK to specify whether or not 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

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 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

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, set the parameter =TCPIP_PROCESS_NAME to the process you want before you start or restart the manager process. For example:

```
ADD DEFINE =TCPIP_PROCESS_NAME, FILE $ZTC3
```

Default None (Do not listen on a port if none is specified)

Syntax `PORT <port num>`

Argument	Description
`<port num>`	The port number.

POSITIONFIRSTRECORD

Valid for Extract

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, ACI's BASE24 Super files).

If you specify SYSKEYCONVERT, Extract turns on POSITIONFIRSTRECORD.

Syntax `POSITIONFIRSTRECORD`

PRIORITY

Valid for Logger

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>`

Argument	Description
`<priority>`	The priority level.

PURGEDATAALTFILES | NOPURGEDATAALTFILE

Valid for Replicat

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

Valid for Manager, Extract, and Replicat

The implementation of this parameter varies, depending on the process.

Extract and Replicat

Use PURGEOLDEXTRACTS in an Extract or Replicat parameter file to delete old trail files whenever 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.

PURGEOLDEXTRACTS should only be used in an Extract or Replicat parameter file if there is only one instance of the process. If multiple groups are reading the same set of trail files, one process could purge a file before another is finished with it. Instead, use the Manager version of PURGEOLDEXTRACTS, which is the preferred use of the parameter in all GoldenGate configurations because it allows you to manage trail files in a centralized fashion.

Default Purge the trail file when moving to the next file in the sequence.

Syntax `PURGEOLDEXTRACTS`

Manager

Use PURGEOLDEXTRACTS in a Manager parameter file to purge trail files when 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 *GoldenGate for HP NonStop Administrator Guide* for recommendations on using the rules.

Default None (do not purge)

Syntax

```
PURGEOLDEXTRACTS <trail name>
[, USECHECKPOINTS | NOUSECHECKPOINTS]
[, <min rule>]
```

Argument	Description
`<trail name>`	The trail to purge. Use the fully qualified name.
`USECHECKPOINTS`	Purges after all Extract or Replicat processes are done 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 <n> Keeps an unmodified file for at least the specified number of hours. ◆ MINKEEPDAYS <n> Keeps an unmodified file for at least the specified number of days. ◆ MINKEEPFILES <n> Keeps at least *n* unmodified files, including the file being considered for purging.

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

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> <group name> [, <purge option>]
```

Argument	Description
<process>	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

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

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 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>]
```

Option	Description
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. GoldenGate recommends you allow the process name to be generated by the system. However, if you must specify an alternative process, PROCESS <process name> allows you to do so.
CPU <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 GoldenGate installation subvolume as indicated in the GGSSUBVOL option.

Example The following example identifies a Reader process to monitor GoldenGate trail AA on \NY.

```
READER EXTTRAIL \NY.$DATA5.GGSDAT.AA, PROCESS $GGRD1, CPU 1, PRI 180
```

READTHRULOCKS | NOREADTHRULOCKS

Valid for Extract

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

Use RECEIVEQWARN to specify a maximum number of messages to hold in queue. GoldenGate issues an EMS warning when the maximum is exceeded.

Default 200

Syntax `RECEIVEQWARN <num messages>`

Argument	Description
`<num messages>`	The message threshold for the queue.

RENAMEBUMPDELAY

VALID FOR CHGNOTE

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 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>

Argument	Description
<seconds>	Can be any value from 1 through 15.

Example RENAMEBUMPDELAY 2

REPERROR

Valid for Replicat

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 HANDLECOLLISIONS parameter on page 193 describes how to handle missing and duplicate record errors as special cases.

See the *GoldenGate for HP NonStop Administrator Guide* "Replicating primary key updates" 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]
[, DELAYSECS]
[, DELAYCSECS]
[, INSERT | UPDATE | DELETE])

REPERROR (<error>, TRANSABORT
[, MAXRETRIES]
[, DELAYSECS]
[, DELAYCSECS])

REPERROR (<error>, FILEOP
[, MAXRETRIES]
[, DELAYSECS]
[, DELAYCSECS]
[, ABEND | WARN | IGNORE]
[, CREATE | ALTER | RENAME | PURGE | SETMODE |
CONTROL | CHANGELABEL])
```

```
REPERROR (<error>, RETRYOPEN
[, MAXRETRIES]
[, DELAYSECS]
[, DELAYCSECS])
```

Argument	Description
`<error>`	Valid values are: DEFAULT, DEFAULT2, or <error number>. ◆ 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. ◆ IGNORE Ignore the error.

Argument	Description
	◆ 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 <n>. 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 page 251. ◆ 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 a DELAY option to delay the retry. The default delay is 60 seconds. 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.

Example 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 (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 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 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"));
```

REPLACEBADCHAR

Valid for Extract, Replicat

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]
```

Argument	Description
<char>	A character value with which to replace unprintable character bytes.
SPACE	Replaces unprintable character values with spaces.

Argument	Description
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. Or... ◆ 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

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]
```

Argument	Description
<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. Or... ◆ When formatting invalid data in ASCII or SQL output, outputs a zero instead.

REPLICAT

Valid for Replicat

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>`

Argument	Description
`<group name>`	The Replicat group name.

Example `REPLICAT FINANCE`

REPNEWCOLUMNS | NOREPNEWCOLUMNS

Valid for Replicat

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

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:mi> |
ON <day of week> |
AT <hh:mm> ON <day of week> |
END
]
```

Arguments	Description
`AT <hh:mi>`	The time of day. Using AT without ON generates a report at the specified time every day.
`ON <day>`	The day of the week. Valid values are SUNDAY through SATURDAY.

Arguments	Description
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

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}
```

Argument	Description
[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.

Example

```
REPORTCOUNT EVERY 5000 RECORDS
```

Example

```
REPORTCOUNT EVERY 10 MINUTES
```

REPORTFILEEXTENTS

Valid for GLOBALS, Extract, Replicat

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 will 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

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>0, <group>1, etc.

Syntax
```
REPORTROLLOVER
{AT <time> |
AT <time> ON <day>}
```

Argument	Description
`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>`	SUNDAY through SATURDAY.

REPORTTMFEXCEPTIONS

Valid for Replicat

Use REPORTTMFEXCEPTIONS for debugging purposes, generally in conjunction with 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)
```

Location ID	Reason
"track duplicate"	A duplicate record that occured 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 dupicate 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

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 may also redirect the SQL log with the define =GGS_REPSQLLOG.

Default APPEND

Syntax REPSQLLOG <file name> [APPEND | PURGE | !]

Argument	Description
<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

Use RESTARTCOLLISIONS to enable HANDLECOLLISIONS until the first checkpoint is done. After the first checkpoint is done RESTARTCOLLISIONS is turned off.

Default NORESTARTCOLLISIONS

Syntax RESTARTCOLLISIONS | NORESTARTCOLLISIONS

RESTARTINTERVAL

Valid for Manager

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 page 228.

Default 20

Syntax RESTARTINTERVAL <minutes>

Argument	Description
<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

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

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 page 242.

Syntax `RETRYDELAY <seconds>`

Argument	Description
`<seconds>`	The number of seconds between retry attempts

RETRYERR

Valid for Extract

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 {<errnum> | DEFAULT | EXPAND}
[, MAXRETRIES <retries>]
[, DELAY <seconds>]
```

Argument	Description
`<errnum>`	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
```

REVERSEWINDOWSECS | REVERSEWINDOWCSECS

Valid for Replicat

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 then applied to the target file, followed by the update, and no error is recorded. If the reverse window expires without a corresponding insert to match the missing update, 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, there will be a message that REVERSEWINDOWSECS is ignored. TMFEXCEPTIONS will still be applied.

Specifying REVERSEWINDOW automatically enables the FIXREVERSEUPDATES parameter.

NOTE REVERSEWINDOWSECS should only be used when the sources is non-TMF audited database. It should not be used when the source a TMF audit trail.

Default ABENDEXPIREDWINDOW

Syntax
```
{REVERSEWINDOWSECS <secs> | REVERSEWINDOWCSECS <csecs>}
[ABENDEXPIREDWINDOW | DISCARDEXPIREDWINDOW]
```

Option	Description
`REVERSEWINDOWSECS <secs> | REVERSEWINDOWCSECS <csecs>`	REVERSEWINDOWSECS <secs> specifies the wait time in seconds. The range is 1 to 30 seconds and the recommended setting is 1. REVERSEWINDOWCSECS <csecs> 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 down time and file inconsistencies can be corrected at a later time.

RMTBATCH

Valid for Extract

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 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 machine and TCP/IP port.

Syntax

```
RMTBATCH <destination file spec>
[, PARAMS <param> [, <param>, ...]];
```

NOTE See also the information on RMTBATCH for QSAM on page 256.

Argument	Description
`<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>`	Can be any one of the following: ◆ AUTOTRUNCATE Executed by Extract or Collector, depending on its placement on the command line.

Argument	Description
	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> 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. FTYPE is used by the collector 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 semi-colon (;).

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, 2006, the file $DATA2.GGS.TL060405 is created, on April 6, $DATA2.GGS.TL060406 is created. On the target system, the following files are created, each containing data from the corresponding source file:

```
/usr/ggsdat/GGS_TL060405
/usr/ggsdat/GGS_TL060406
```

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:

Parameter	Description
`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 page 285.

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 SuperExtract output files, but may 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.

Parameter	Description
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

Use RMTFILE when the remote 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 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>, <maxextents>)]
[, MAXFILES <num files>]
[, MEGABYTES <num megabytes>]
[, OWNER <group num>, <user num>]
[, SECURE “<rwep>”]
```

Argument	Description
`<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>, <maxextents>)`	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 num>, <user num>`	Determines which user group owns the RMTFILE.

Argument	Description
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

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.

Two methods exist for specifying RMTHOST:

- Static Collector identification, see "Identifying a static collector" on page 259.
- Dynamic Collector identification, see "Identifying a dynamic collector" on page 260.

You must also specify the TCPIPPROCESSNAME parameter when the NonStop TCP/IP process name is not $ZTC0 (see the example for the RMTFILE parameter).

Identifying a static collector

When using the static collector method, the target machine must be running the Collector program before starting Extract.

Syntax

```
RMTHOST {<DNS host name> | <IP address>} [, PORT <port num>]
```

Argument	Description
<DNS host name>	The DNS host name.
<IP address>	The IP address.
PORT <port num>	An optional argument specifying the port for Collector. The default port is 7819.

Identifying a dynamic collector

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>}, MGRPORT <port num>
              [, <options>]
```

Argument	Description
`<host name>`	The remote host name. If <host name> is specified, do not specify <IP address>.
`<IP address>`	The remote host IP address. If <IP address> is specified, do not specify <host name>.
`MGRPORT <port num>`	DEFAULT or the port number on which Manager is listening. The default port number is 7809. Manager dynamically starts the Collector. Do not specify a Collector port when Manager has dynamically started the Collector.
`<options>`	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 Golden Gate Software. <type> is one of: NONE, GGS, or BLOWFISH. NONE specifies no encryption. GGS specifies a default key generated by GoldenGate. BLOWFISH specifies a user-defined key. The destination Collector must be version 5.40 or above.

Argument	Description
	◆ 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" on page 282. 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 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.

RMTHOSTALT

Valid for Extract

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 num>
```

Argument	Description
`<host name>`	The host name. If <IP address> is specified, do not specify <host name>.
`<IP address>`	The IP address. If <host name> is specified, do not specify <IP address>.

Argument	Description
`<port num>`	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.

Example The following example causes data to be output over TCP/IP process $ZTC0 to the system designated by 193.2.23.1 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 193.2.23.2 on port 7831.

```
TCPSWITCHERRS 3
TCPIPPROCESSNAME $ZTC0
TCPIPPROCESSNAME $ZTC1
RMTHOST 193.2.23.1, PORT 7830
RMTHOSTALT 193.2.23.2, PORT 7831
RMTTRAIL $DATA5.GGSDAT.RT
```

RMTTASK

Valid for Extract

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>”]
```

Argument	Description
`<task name>`	Enter REPLICAT, currently the only task on the remote system.
`GROUP <replicat group>`	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 runtime 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

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 GoldenGate trails, refer to page 58.

Syntax `RMTTRAIL <trail name>`

Argument	Description
`<trail name>`	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 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 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 123.324.34.21, MGRPORT 7809
RMTTRAIL /ggs/dirdat/XX
FILE $DATA2.FINANCE.ACCOUNTS;
RMTTRAIL /ggs/dirdat/YY
FILE $DATA3.FINANCE.ORDERS;
```

ROLLOVER

Valid for Extract

Use ROLLOVER to specify times at which local or remote trails automatically roll over, in sequence. Each ROLLOVER rule applies to all 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:mi>]
[ON <day of week>]
[AT <hh:mm> ON <day of week>]
[AT END]
[REPORT]
```

Arguments	Description
`AT <hh:mi>`	The time of day. Using AT without ON generates a report at the specified time every day.
`ON <day>`	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 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

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.

Default None

Syntax `SETENV (<environment variable>)`

Argument	Description
`<environment variable>`	The name of the environment variable

Example
```
EXTRACT GGS01
SETENV (TRAIL1=$DATA01.LOGGER.A1)
SETENV (TRAIL2=$DATA01.LOGGER.A2)
FILE $*.*.*;
EXTTRAIL ?TRAIL1
FILE $*.*.*
EXTTRAIL ?TRAIL2
```

SHORTREADDELAY

Valid for Extract

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>`

Argument	Description
`<seconds>`	The number of seconds to delay.

SOURCEDEFS

Valid for Extract, Replicat

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>`

Argument	Description
`<file name>`	The name of the file containing the data definitions.

SOURCEISFILE | SOURCEISTABLE

Valid for Extract

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.

When specifying SOURCEISFILE or SOURCEISTABLE, omit EXTRACT and SPECIALRUN.

Syntax
```
SOURCEISFILE | SOURCEISTABLE
[, USEKEY]
[, SELECTVIEW]
[, FASTUNLOAD | FASTUNLOADSHARED]
```

Argument	Description
USEKEY	Use any WHERE or FILTER clauses designated as part of all FILE or TABLE entries as criteria when selecting data from the source table. USEKEY 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 USEKEY 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).

Argument	Description
	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.
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.

SPECIALRUN

Valid for Extract, Replicat

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 GoldenGate or TMF audit trails; a Replicat from GoldenGate trails.

Extract

You can specify EXTTRAILSOURCE or EXTFILESOURCE for Extract to read data from 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>]
```

Argument	Description
EXTTRAILSOURCE <trail name>	The name of the GoldenGate trail to use as the data source.
EXTFILESOURCE <trail file name>	The name of the GoldenGate trail containing the single file to use as the data source.

Example `SPECIALRUN EXTTRAILSOURCE $data1.ggsdat.et`

Example `SPECIALRUN EXTFILESOURCE $data3.etdat.datfile1`

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;
```

STATOPTIONS

Valid for Extract, Replicat

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] (Replicat only)
```

Argument	Description
`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.

Example 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 Sample results for STATOPTIONS ZEROSUPPRESS:

```
Total # records written to RFUNLDS 349
   \NY015.$DATA06.QABASE.BRANCH    # inserts: 100
   \NY015.$DATA07.LQASRC.HISTORY   # inserts: 249
```

Example 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
```

SUPPRESSALTERMESSAGES | NOSUPPRESSALTERMESSAGES

Valid for Extract

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 numerious 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

SUPPRESSFILEOPMESSAGES | NOSUPPRESSFILEOPMESSAGES

Valid for Extract, Replicat

Use SUPPRESSFILEOPMESSAGES to suppress the output of messages generated after FILE RENAME, PURGE, CREATE, ALTER, SETMODE.

Syntax NOSUPPRESSFILEOPMESSAGES

SUPPRESSMARKERMESSAGES

Valid for GLOBALS, Extract, Replicat

Use SUPPRESSMARKERMESSAGES to suppres 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

Use SWAPVOL to designate a swap volume for processes created by 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 GoldenGate is installed

Syntax `SWAPVOL <volume>`

Argument	Description
`<volume>`	Specifies the swap volume.

SYNCFILE

Valid for Syncfile

Use SYNCFILE when the parameter file has a different name than the group name.

Syntax `Syncfile <group name>`

Argument	Description
`<group name>`	The Syncfile group name.

SYSKEYCONVERT

Valid for Extract

Use SYSKEYCONVERT for direct file extraction only. SYSKEYCONVERT specifies the format of the syskey in the 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, ACI's 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}`

Argument	Description
`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 ACI's 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

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 page 162.

TABLEEXCLUDE

Valid for Extract

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>`

Argument	Description
`<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

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`

TCPFLUSHBYTES

Valid for Extract

Use TCPFLUSHBYTES to determine the size of the record buffer that Extract sends to remote hosts. Depending on the remote system, changing this parameter can have a significant impact on performance. Trial and error is usually the best way to discover an optimal setting.

Default 28000

Syntax `TCPFLUSHBYTES <bytes>`

Argument	Description
`<bytes>`	The buffer size in bytes.

TCPIPPROCESSNAME

Valid for Manager, Extract

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.

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 page 261 for an example.

Default `$ZTC0`

Syntax `TCPIPPROCESSNAME <process>`

Argument	Description
`<process>`	The TCP/IP process name.

Example The following Extract parameter setting will cause data to be output over TCP/IP process `$ZTC4` instead of the default `$ZTC0`.

```
TCPIPPROCESSNAME $ZTC4
```

TCPIPSWITCHERRS

Valid for Extract

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 of retries>
```

Argument	Description
`<number of retries>`	The number of retries.

Refer to RMTHOSTALT on page 261 for an example using TCPIPSWTICHERRS.

TCPSOURCETIMER | NOTCPSOURCETIMER

Valid for Extract

Use TCPSOURCETIMER or NOTCPSOURCETIMER to specify whether or not 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

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 page 140 for a description of the DISKTHRESHOLD parameter and audit threshold processing.

Default `THRESHOLD 20`

Syntax `THRESHOLD <percent left> | NOTHRESHOLD`

Argument	Description
`<percent left>`	The specified percentage.

TMFDUMPAGE

Valid for GLOBALS, Manager

TMFDUMPAGE limits information returned by the TMFDUMPINFO command to dumps that are created during the specified number of days.

Default 30

Syntax `TMFDUMPAGE <num days>`

Argument	Description
`<num days>`	The number of days for limiting TMF dump information.

TMFDUMPTABLEENTRIES

Valid for GLOBALS, Manager

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

Default 1024

Syntax `TMFDUMPTABLEENTRIES <max dumps>`

Argument	Description
`<max dumps>`	The number of dump entries to display. The maximum allowed is 6000.

TMFEXCEPTIONS | NOTMFEXCEPTIONS

Valid for Replicat

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

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 GoldenGate TMF commands in GGSCI (see page 67).

Default 15

Syntax TMFREFRESHINTERVAL <minutes>

Argument	Description
<minutes>	The refresh interval in minutes.

TMFTRAILTRACE

Valid for Extract

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

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 a

GoldenGate support session.

Syntax TRACE

TRACEALLOPENS

Valid for Logger

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

TRACECLOSES

Valid for Logger

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

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

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

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

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

Use UPDATEINSERTS to change insert operations in the target to update operations.

Default `NOUPDATEINSERTS`

Syntax `UPDATEINSERTS | NOUPDATEINSERTS`

UPREPORT

Valid for Manager

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>`

Argument	Description
<minutes>	The reporting interval in minutes.
<hours>	The reporting interval in hours.

VERBOSE

Valid for Syncfile

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;
```

VERSIONERR

Valid for Extract

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 allows you to override these defaults by specifying custom responses to old, new and out-of-sync records.

Syntax `VERSIONERR <type>, {CONTINUE | WARN | DISCARD | ABEND}`

Argument	Description

`<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`

WAITFILEEVENT

Valid for Extract, Replicat

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>, 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>]
```

Argument	Description
`<source filespec>`	The file name or wildcard name to which the wait rules apply.

Argument	Description
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.

Example 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 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"
```

WARNRATE

Valid for Replicat

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>`

Argument	Description
`<num errors>`	The error report rate.

Y2KCENTURYADJUSTMENT | NOY2KCENTURYADJUSTMENT

Valid for Extract, Replicat

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 corrupts 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`

CHAPTER 3

Collector Parameters

This chapter describes Collector parameters you can use to control and modify Collector operations. Collector parameters are specified by a dash (-) followed by a letter and optional arguments. 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 the *Reference Guide*.

Parameter	Description
`-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.

Parameter	Description
`-d <NonStop definitions file>`	Specifies a local file that contains exported definitions of NonStop tables and files. Use this parameter whenever the Extract program performs ASCII formatting, unless the FORMATLOCAL Extract parameter is specified. The source definitions file must contain a definition for each source table for which ASCII conversion occurs. You may specify multiple definitions.
`-DEBUG`	Starts the Collector in DEBUG mode to allow the user to analyze the actions of the program code.
`-e <version error type> <action>`	Directs the Collector to respond to specific formatting conditions in custom ways. Default values are almost always sufficient. <version error type> can be one of the following: ◆ NEW Checks for records that contain more data than anticipated (more columns than the current definition). The Collector process may need an updated version of the source table (DEFGEN may need to be run again). The default action is ABEND. ◆ OLD Checks for records that contain less data than anticipated. This usually indicates that a record has fewer columns that the table's current definition, which is considered a normal condition. The default action is CONTINUE. ◆ OUTOFSYNC Checks for records that can not be converted according to the definition provided. The default action is ABEND. <action> can be one of the following: ◆ ABEND Discards the record and directs the Extract process to end immediately. ◆ CONTINUE Processes the record (if possible) regardless of the conversion error encountered. ◆ DISCARD Outputs the record to a discard file (if one is specified with the -x). Collector sends a warning to the error file for the first discarded record and continues to process records. To specify more than one error type, use -e multiple times, such as: `-e OLD CONTINUE -e NEW DISCARD.`

Parameter	Description
-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 GoldenGate security, see the *GoldenGate for HP NonStop Administrator Guide*.
-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 GoldenGate's performance.
-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 GoldenGate security, see the *GoldenGate for HP NonStop Administrator Guide*.
-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.
-o SHARED	Opens files as shared rather than protected. Specify this parameter if using more than one collector in a fault-tolerant configuration. **Note:** If BULKIO or BULKIOLEN parameters are used, -o is ignored. See page 261 for details about the Extract RMTHOSTALT parameter.
-p <port number>	A TCP/IP port number on which the Collector listens for connections from the Extract process. The default is port 7819.

Parameter	Description
`-R <number> | NULL | UNPRINTABLE`	Specifies a substitute value for ASCII formatted data to be used when invalid numeric data is encountered. The default is 0. Valid values are: ◆ <number> specifies a substitute number. ◆ NULL specifies NULL if the target column accepts NULL values; otherwise replaces with zero. ◆ UNPRINTABLE rejects any column with unprintable data. The process stops and reports the bad value This parameter performs the same function as the REPLACEBADNUM parameter (see page 246).
`-r <character> | SPACE | NULL | UNPRINTABLE | NONE`	Specifies a substitute character for ASCII formatted data to be used when an unprintable character is encountered. The default is SPACE. NONE specifies no replacement and can be used to suppress transformation of double-byte character set values to default characters, when the incoming data is converted to ASCII format. This parameter performs the same function as the REPLACEBADCHAR Extract parameter (see page 245).
`-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 GoldenGate Support.
`-TCPSTATS`	Specifies that TCP statistics should be printed at file rollover.
`-v`	Specifies that NonStop Collector processes use EMS logging.
`-x <discard file>`	Specifies a discard file for outputting records that could not be formatted.

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>, ...];
```

Where:

- <file spec> is 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> is one of those explained in the following table.

Option	Description
`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`	Allows you to 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> [, <serial number>, ...]`	Mandatory parameter. Identifies the serial numbers of the volumes on which the target data set will reside. Specify up to 255 volume serial numbers.

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 * %";
```

CHAPTER 4

Field Conversion Functions

Using field conversion functions,you can manipulate numbers, strings and source columns or field values into the appropriate format for target columns. GoldenGate supplies field conversion functions for:

- 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 GoldenGate functions for the NonStop platform.

Function syntax

Field conversion functions use the following general syntax:

```
@FUNCTION (expression)
```

Function element	Description
`@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 GoldenGate functions for NonStop, based on their functionality. An alphabetized reference follows on page 291.

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.
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.

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.

Function	Description
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.
NUMBIN	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 GoldenGate environment.

Alphabetical reference

This begins the alphabetical reference for GoldenGate functions on the NonStop platform.

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>])
```

Argument	Description
<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.

Example 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 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”)
```

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)
```

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>] [, ...])
```

Argument	Description
`<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.

Example 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 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>)
```

Argument	Description
<value>	One or more values on which you are performing calculations.
<operator>	A valid arithmetic or logical operator.

Example This example adds AMT and AMT2 and returns the total to AMOUNT_TOTAL.

```
AMOUNT_TOTAL = @COMPUTE (AMT + AMT2)
```

Example This example is invalid because the expression is not enclosed in parentheses.

```
AMOUNT_TOTAL = AMT + AMT2
```

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>] [, ...])
```

Argument	Description
<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" on page 294 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" on page 294 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.
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 page 295.
JUL	Julian day
TTS	NonStop 48-bit timestamp
PHAMIS	PHAMIS application date format
FFFFFF	Fraction (up to microseconds)

Descriptor	Description
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>)
```

Argument	Description
<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.

Example To calculate the number of days since the beginning of the year 2000:

```
differential = (@DATEDIFF("DD","2000-01-01",@DATENOW()))
```

Example 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)
```

DATENOW

@DATENOW takes no arguments and returns the current date and time in the format YYYY-MM-DD HH:MI:SS. The date and time are returned in local time, including adjustments for daylight savings time.

Syntax
```
@DATENOW ()
```

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>])
```

Argument	Description
`<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.

Example 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 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 GoldenGate processing, including lag information, the last replicated operation, and GoldenGate environment information.

Syntax

```
@GETENV (<info type>)
```

Info type	Description
`"COMMITTIMESTAMP"`	Returns the timestamp when the transaction was committed as an interger representing the Julian GMT.
`"JULIANTIMESTAMP"`	Returns the current Julian GMT timestamp in the form of an interger such as 211919385876765584.
`"LOCALTIMESTAMP"`	Returns the current system time as an interger representing the Julian LCT.
`("LAG", "<unit>")`	Returns lag information. See "Reporting lag information" on page 298.
`("LASTERR","<option>")`	Returns information about the last replicated operation, including detailed error information. See "Returning information from Replicat" on page 298.

Info type	Description
(“GGENVIRONMENT”, “<option>”)	Returns GoldenGate environment information. See “Returning GoldenGate environment information” on page 299.
(“GGFILEHEADER”, “<option>”)	Returns the format and properties of a GoldenGate trail file, which is stored in the file header record sent from open systems.
(“GGHEADER”, “option>”)	Returns GoldenGate record header information. See “Returning record header information” on page 299.
“RECSOUTPUT”	Returns the total number of records processed.
("TLFKEY", SYSKEY <unique key>)	Enables a unique key to be associated with TLF/PTLF records in ACI's BASE24 application. For more information on these associations see page 305

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	Description
“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	Description
“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.

Option	Description
"ERRTYPE"	Returns the type of error. Possible results are: ◆ DB (for database errors) ◆ MAP (for errors in mapping before replicating the record).

Returning GoldenGate environment information

Use the GGENVIRONMENT option of @GETENV to return information about the GoldenGate environment. This option is valid for the Extract and Replicat program processing.

Syntax

```
@GETENV ("GGENVIRONMENT", "<option>")
```

Option	Description
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	Description
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 timezone as a string in the format of YYYY-MM-DD HH:MI:SS.FFFFFF, for example: 2007-01-24 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.

Option	Description
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 Appendix 3 in the *GoldenGate for HP NonStop Administrator Guide*.
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 a GoldenGate extract file or trail file that are stored in the file header sent from a 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 a GoldenGate process to determine whether the records are in a format that the current version of GoldenGate supports.

The trail header fields are stored as tokens, where the token format remains the same across all versions of GoldenGate. If a version of 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 GoldenGate.

This option is valid for the Replicat process. Both GGFILEHEADER and <environment value> must be enclosed within double quotes.

NOTE If a given database, operating system, or GoldenGate version does not provide information that relates to a given token, a NULL value will be returned.

Syntax `@GETENV ("GGFILEHEADER", "<return_value>")`

Environment value	Return value
Information about the trail file.	
"COMPATIBILITY"	Returns the GoldenGate compatibility level of the trail file. The compatibility level of the current 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 GoldenGate version 10.0 or later, which supports file headers that contain file versioning information. ◆ 0 means that the trail file is of a GoldenGate version that is older than 10.0. File headers are not supported in those releases. The 0 value is used for backward compatibility to those GoldenGate versions.
"CHARSET"	Returns the global character set of the trail file. For example: WCP1252-1
"CREATETIMESTAMP"	Returns the time that the trail was created, in local GMT Julian time in INT64.
"URI"	Returns the universal resource identifier of the process that created the trail file, in the format of: <host_name>:<dir>:[:<dir>][:<dir_n>]<group_name> **Where:** ◆ host_name is the name of the server that hosts the process ◆ dir is a subdirectory of the 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"	Returns a 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"	Returns the name of the trail file. Can be absolute or relative path, with forward or backward slash depending on the filesystem.

Environment value	Return value
"FILEISTRAIL"	Returns a 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"	Returns the sequence number of the trail file, without any leading zeros. For example, if a file sequence number is aa000026, FILESEQNO returns 26.
"FILESIZE"	Returns the 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"	Returns 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"	Returns the time that the first record was written to the trail file. Value is NULL until the trail file is completed.
"LASTRECIOTIME"	Returns the time that the last record was written to the trail file. Value is NULL until the trail file is completed.
Information about the GoldenGate process thata created the trail file.	
"GROUPNAME"	Returns the name of the group 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, "ggext."
"DATASOURCE"	Returns the data source that was read by the process. Can be one of: ◆ DS_EXTRACT_TRAILS (source was a GoldenGate extract file, populated with change data) ◆ DS_LOG_TABLE (source was a 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"	Returns the major version of the Extract process that created the trail, expressed as an integer. For example, if a version is 1.2.3, it returns 1.

Environment value	Return value
"GGMINORVERSION"	Returns the minor version of the Extract process that created the trail, expressed as an integer. For example, if a version is 1.2.3, it returns 2.
"GGVERSIONSTRING"	Returns the maintenance (or patch) level of the Extract process that created the trail, expressed as an integer. For example, if a version is 1.2.3, it returns 3.
"GGMAINTENANCELEVEL"	Returns the maintenance version of the process (xx.xx.xx).
"GGBUGFIXLEVEL"	Returns the patch version of the process (xx.xx.xx.xx).
"GGBUILDNUMBER"	Returns the build number of the process.
Information about the local host of the trail file.	
"HOSTNAME"	Returns the DNS name of the machine where the Extract that wrote the trail is running. For example: ◆ sysa ◆ sysb ◆ paris ◆ hq25
"OSVERSION"	Returns the major version of the operating system of the machine where the Extract that wrote the trail is running. For example: ◆ Versions10_69 ◆ #1 SMP Fri Feb 24 16:56:28 EST 2006 ◆ 5.00.2195 Service Pack 4
"OSRELEASE"	Returns the release version of the operating system of the machine 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"	Returns the type of operating system of the machine where the Extract that wrote the trail is running. For example: ◆ SunOS ◆ Linux ◆ Microsoft Windows
"HARDWARETYPE"	Returns the type of hardware of the machine where the Extract that wrote the trail is running. For example: ◆ sun4u ◆ x86_64 ◆ x86

Environment value	Return value
Information about the database that produced the data in the trail file.	
"DBTYPE"	Returns the type of database that produced the data in the trail file. Can be one of: DB2 UDB DB2 ZOS CTREE INGRES MSSQL MYSQL ORACLE SQLMX SYBASE TERADATA NONSTOP ENSCRIBE ODBC
"DBNAME"	Returns the name of the database, for example findb.
"DBINSTANCE"	Returns the name of the database instance, if applicable to the database type, for example ORA1022A.
"DBCHARSET"	Returns 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"	Returns the major version of the database that produced the data in the trail file.
"DBMINORVERSION"	Returns the minor version of the database that produced the data in the trail file.
"DBVERSIONSTRING"	Returns the maintenance (patch) level of the database that produced the data in the trail file.
"DBCLIENTCHARSET"	Returns the character set that is used by the database client.
"DBCLIENTVERSIONSTRING"	Returns the maintenance (patch) level of the database client. (For some databases, this will be empty.)
Recovery information carried over from the previous trail file.	
"RECOVERYMODE"	Returns recovery information for internal GoldenGate use.
"LASTCOMPLETECSN"	Returns recovery information for internal GoldenGate use.
"LASTCOMPLETEXIDS"	Returns recovery information for internal GoldenGate use.
"LASTCSN"	Returns recovery information for internal GoldenGate use.

Environment value	Return value
"LASTXiD"	Returns recovery information for internal GoldenGate use.
"LASTCSNTS"	Returns recovery information for internal GoldenGate use.

Associating BASE24 keys and records

Use the TLFKEY option of @GETENV to associate a unique key with TLF/PTLF records in ACI's 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>)
```

Option	Description
<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>)
```

Argument	Description
<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>])
```

Argument	Description
<length>	Optional. Specifies the binary output length in bytes. The maximum value of <length> is the length of the target column.

Example 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 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>, <value if expression is non-zero>,
<value if expression is zero>)
```

Argument	Description
`<conditional expression>`	The conditional expression.
`<value if expression is non-zero>`	A non-zero result is considered TRUE.
`<value if expression is zero>`	A zero result is considered FALSE.

Example The following returns AMT only if AMT is greater than zero, otherwise zero is returned.

```
AMOUNT_COL = @IF (AMT <= 0, 0, AMT)
```

Example The following returns WEST if STATE is CA, AZ or NV, otherwise returns EAST.

```
REGION = @IF (@VALONEOF (STATE, "CA", "AZ", "NV"), "WEST", "EAST")
```

Example 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 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 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 (<character column or 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. 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 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>] [, ...]))
```

Argument	Description
<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>.

Argument	Description
<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, GoldenGate calculates ranges based on the table's primary key.

Example 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 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, 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 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 (<str1>, <str2> [,...])
```

Argument	Description
<str1>	The first string to be concatenated.
<str2>	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>)
```

Argument	Description
<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 (str1, str2)`

Argument	Description
`<str1>`	The first string to compare.
`<str2>`	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>)`

Argument	Description
`<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>])`

Argument	Description
`<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.

Argument	Description
"<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>)
```

Argument	Description
<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>)
```

Argument	Description
<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>, ...])
```

Argument	Description
`<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 (<character column or literal string>,
<character column or literal string>,
<max compare length>)
```

Argument	Description
`<character column or literal string>`	The data to compare.
`<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>])
```

Argument	Description
`<field>`	The name of a source numeric field.
`LEFT`	Left justify, fill the rest of the target column with spaces

Argument	Description
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 preceeding 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>)
```

Argument	Description
<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 column>, <search string>, <substitute string>, ...)`

Argument	Description
`<source string or column>`	The string or column to replace.
`<search string>`	The value to be replaced.
`<substitute string>`	The replacement value.

Example 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 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>)`

Argument	Description
`<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>)`

Argument	Description
`<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 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 GoldenGate macroor user exit.

Syntax

```
@TOKEN ("<token name>")
```

Argument	Description
"<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")
```

CHAPTER 5

DEFGEN Arguments

This chapter describes the runtime arguments for the DEFGEN utility. You enter the arguments 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 the *GoldenGate for HP NonStop Administrator Guide*.

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.
RECORDNAMEPROMPTING	Supplies the name of an existing record definition to use when generating a definition for a new table.

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> [, ...]`

Argument	Description
`<separator>`	A character separator for defining array output. See "Defining a separator" on page 318.
`<options>`	For information about the options, see: ◆ Information on inheriting OCCURS on page 319. ◆ "Resolving duplicate field names" on page 319. ◆ "Omitting or including redefined fields" on page 320. ◆ "Fixing long field names" on page 320. ◆ "Altering field array display" on page 321.

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	Description
USEBRACKETS	Use brackets [ ].
USEDASH	Use a dash -.
USEUNDERSCORE	Use an underscore _.
USETWOUNDERSCORES	Use two underscores __.

Separator	Description
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>

Option	Description
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.

Example 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 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]

Argument	Description
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]]`

Argument	Description
`<length>`	The field name length.
`NOFIXLONGNAMES`	Specify to prevent prompting for a new column name. A warning message is issued.

Default 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}`

Argument	Description
`<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.

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.

Default File name prompting

Syntax `RECORDNAMEPROMPTING`

CHAPTER 6

DDLGEN Arguments

This chapter describes the runtime arguments for the DDLGEN utility. You enter the arguments at the command line when you run the utility. DDLGEN provides instructions for generating table definitions for target databases based on existing Enscribe and SQL definitions. For more information about using DDLGEN, see the *GoldenGate for HP NonStop Administrator Guide*.

Run-time arguments

Syntax

```
TACL> RUN DDLGEN /IN [command file]/ [-d <file name>]
```

The DDLGEN syntax shown above includes the following run-time arguments:

Argument	Description
`/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>.

Template parameters

GoldenGate includes a template for each supported database. The following arguments can be used in these templates. For more information about using templates, see the *GoldenGate for HP NonStop Administrator Guide*.

Parameters	Description
OMITNULL | 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.

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

CHAPTER 7

MEASFLS and MEASRPT Parameters

Use the MEASFLS and MEASRPT programs to assist with the sizing of GoldenGate trails.

- MEASFLS parameters collect statistics using the Measure program.
- MEASRPT parameters interpret the statistics collected by MEASFLS and produce a report to assist with sizing.

Parameter summaries

The following are summaries of the MEASFLS and MEASRPT parameters. For alphabetic reference sections, see:

- MEASFLS parameters on page 328.
- MEASRPT parameters on page 332.

MEASFLS parameters summary

MEASFLS starts Measure processes for collecting file I/O data at specified intervals. The collected data is used by MEASRPT to generate a report. See the *GoldenGate for HP NonStop Administrator Guide* for details about creating the MEASFLS parameter file and running the program.

Parameter	Description
DURATION	Determines how long the measurement runs.
FILE	Specifies a file name or wildcard file set.
FILEMEGABYTES	Enables the explicit sizing of Measure files.
GETSYSTEMUTILS	Specifies whether TMF statistics are collected for programs that reside on $SYSTEM.
GETTMFDETAIL	Specifies the collection of TMF begin and abort transaction statistics on a per-process basis.
INTERVAL	Specifies a reporting interval for gathering statistics.

Parameter	Description
MAXFILESPERMEAS	Increases the number of files per measure.
MEASFILES	Identifies the location for the files that Measure uses to gather statistics.
PURGEMEASFILES	Controls whether existing measurement files are purged before starting Measure (default is OFF).
START	Controls when the measurement starts.
STOP	Controls when the measurement stops.
WILDCARD	Specifies a name of a file or wildcard file set.

MEASRPTS parameters summary

MEASRPT outputs GoldenGate sizing statistics gathered by MEASFLS. See the *GoldenGate for HP NonStop Administrator Guide* for details about creating the MEASFLS parameter file and running the program.

Parameter	Description
AVGCOMPRESSEDBYTES	Estimates the bytes needed per update record for bandwidth and disk storage.
DEFAULTEXCLUDES	Determines whether or not I/O resulting from OSIMAGE and ORSERV (FUP RELOAD) is excluded in the report.
DURATION	Determines the reporting period.
EXCLUDEALTKEYS	Determines whether or not statistics for alternate key files are included in the report.
EXCLUDEFILE	Determines whether or not statistics for a file set are included in the report.
EXCLUDEPROGRAM	Determines whether or not statistics for files updated by the specified programs are excluded from the report.
EXCLUDETANDEMFILES	Determines whether or not statistics for files with file codes between 1 and 1000 are excluded.
FILEDETAIL	Determines whether or not the details of each file's activity are reported by reporting period.
GETAUDITED	Determines whether or not audited files are included in the report.
GETENSCRIBE	Determines whether or not Enscribe files are included in the report.

Parameter	Description
GETNONAUDITED	Determines whether or not non-audited files are included in the report.
GETSQL	Determines whether or not NonStop SQL tables are included in the report.
GETTMFDETAIL	Determines whether or not TMF transaction information is reported on a per-program basis.
INCLUDEFILE	Determines whether or not statistics for a file set are included in the report.
INCLUDEPROGRAM	Determines whether or not statistics for files updated by the programs specified are included in the report.
INTERVAL	Specifies an interval for reporting peak activity.
LISTLIMIT	Limits the number of files and programs listed in the report to those files or programs that generate the most output activity.
MEASFH	Identifies the location of the MEASFH program.
MEASFILES	Identifies the location of the files Measure will use to report statistics.
PROGDETAIL	Determines whether or not to report details regarding each program that updates the specified files.
PROGSTATS	Determines whether or not to report statistics for programs that update the file set specified.
REPORTRATE	Determines whether or not statistics are reported as absolute numbers or on a per-second basis.
RETENTION	Estimates the maximum disk file capacity needed for an extended outage.
START	Controls when the reporting period starts.
STOP	Controls when the reporting period stops.
SUPPRESSZEROTOTALS	Determines whether or not to report files for which statistics were captured but did not experience file updates.

MEASFLS parameter reference

DURATION

Use DURATION to specify a length of time for collecting measurements in terms of minutes, hours, or days. The following is an example:

```
DURATION 48 HOURS
```

It starts measurements at the time specified by the START parameter, and continues until 48 hours have elapsed.

Alternatively, you can use the STOP parameter to specify a time, such as 15:00:00, to stop measurements.

Default One day

Syntax `DURATION <number> {MINUTES | HOURS | DAYS}`

Argument	Description
`<number>`	The amount of time for collecting measurements.
`MINUTES | HOURS | DAYS`	The unit of time for collecting data.

FILE

Use FILE to specify a file set to collect measured data. You can include multiple FILE statements or use wildcard arguments. However, when using wildcards, the arguments are expanded, creating an entry for each file on the system that satisfies the wildcard definition.

For example, FILE $DATA3.SUB1.ACCOUNT explicitly names the file set. Alternatively, FILE $DATA*.SUB?.FL2 uses a wildcard argument to specify any file set in any volume beginning with $DATA, and any subvolume with the name SUBn, that contains a file set FL2.

Whenever possible, use the WILDCARD parameter as an alternative to wildcard arguments to reduce the chances of collecting unnecessary measurements. However, be aware that FILE gives you more flexibility for defining file sets than does WILDCARD.

For example, use FILE to explicitly specify secondary partitions you wish to measure block-level activity against, such as from FUP LOAD. Using FILE, you can also filter out activity against selected files and types of files during the reporting phase. Explicitly specifying the files that might be accessed during data gathering allows more flexible reporting.

Syntax `FILE <$vol.subvol.fileset>`

Argument	Description
`<$vol.subvol.fileset>`	The fully qualified file set name.

FILEMEGABYTES

Use FILEMEGABYTES to set the size of Measure files.

Syntax `FILEMEGABYTES <megabytes>`

Argument	Description
`<megabytes>`	The file size, in megabytes.

GETSYSTEMUTILS

Use GETSYSTEMUTILS to specify whether or not TMF statistics are collected for programs that reside on $SYSTEM (notably SQLCI).

Default `OFF` (Do not collect TMF statistics)

Syntax `GETSYSTEMUTILS {ON | OFF}`

GETTMFDETAIL

Use GETTMFDETAIL to specify whether or not TMF statistics are collected on a per-process basis.

Default `OFF` (Do not collect TMF statistics)

Syntax `GETTMFDETAIL {ON | OFF}`

INTERVAL

Use INTERVAL to specify a statistics reporting interval in terms of the amount of minutes, hours, or days that have elapsed. For example, INTERVAL 30 MINUTES, reports performance statistics every 30 minutes.

You can specify short reporting intervals for MEASFLS, then longer intervals for MEASRPT (using the MEASRPT INTERVAL parameter). Since short intervals can consume significant amounts of disk space and CPU, intervals of at least 30 minutes are recommended and should suffice for most installations. INTERVAL determines the level that peak activity can be reported against.

Default One hour

Syntax `INTERVAL <num units> <unit>`

Argument	Description
`<num units>`	The amount of time for collecting data.
`<unit>`	The interval for collecting data. Enter one of the following literal values: ◆ MINUTES ◆ HOUR ◆ DAY

MAXFILESPERMEAS

Use MAXFILESPERMEAS to change the default maximum number of files per measure.

Default 40

Syntax `MAXFILESPERMEAS <num files>`

Argument	Description
`<num files>`	The number of files per measure. The maximum is 40,000.

MEASFILES

Use MEASFILES to identify the locations of the file sets into which Measure gathers statistics. The MEASFLS program creates one measurement, and one Measure file, for every 40 file entries created with the FILE parameter. If a FILE entry expands to 120 files, for example, three Measure files are required.

If you specify file locations with the WILDCARD parameter, only one file set argument is required since MEASFLS relies on Measure to control the expansion.

If you are specifying PURGEMEASFILES ON, MEASFILES must follow PURGEMEASFILES.

Syntax `MEASFILES <$vol.subvol.fileset>`

Argument	Description
`<$vol.subvol.fileset>`	A fully qualified file set name with the file portion no more than six characters. MEASFLS appends a sequence number to the MEASFILES entry for each measurement

Example
```
MEASFILES $DATA3.MEASDAT.MDAT
                WILDCARD $*.DAT*
```

PURGEMEASFILES

Use PURGEMEASFILES to control whether or not existing measurement files are purged before starting the Measure program.

Must precede the MEASFILES parameter.

Default `OFF` (Do not purge measurement files)

Syntax `PURGEMEASFILES {ON | OFF}`

START

Use START to define when the measurements begin. The default is the date and current time. For example, START 2006-03-12 15:00:00, specifies measurements to start on March 12, 2006 at 1500 hours.

By default, the current time is rounded up to an even unit of time. For example, if the current time is 9:52, and the INTERVAL parameter specifies 1 hour for the <time> argument, the measurement starts at 10:00.

Default The current date and time

Syntax `START <date> <time>`

Argument	Description
`<date>`	The date in yyyy-mm-dd format. The default is the current date.
`<time>`	The time is hh:mm:ss format. The default is the current time, rounded to the next hour. When specifying a time, specify 00 (zeros) for minutes and seconds.

STOP

Use STOP instead of the DURATION and START parameters for specifying a time of day to stop measurements. For example, STOP 2006-03-12 15:00:00 specifies measurements to stop on March 12, 2006 at 1500 hours.

Alternatively, you can use the DURATION and START parameter.

The default is the start time defined with the START parameter plus the default value for the DURATION parameter (one day). The STOP time is rounded up to an even interval. For example, if the stop time is 13:45, and the INTERVAL parameter specifies one hour, the measurement stops at 14:00.

Syntax `STOP <date> <time>`

Argument	Description
`<date>`	The date in yyyy-mm-dd format. The default is the current date.
`<time>`	The time is hh:mm:ss format. The default is the current time, rounded to the next hour. When specifying a time, specify 00 (zeros) for minutes and seconds.

WILDCARD

Use the WILDCARD parameter instead of FILE or MEASFILES to let the Measure program determine the files and file sets to configure and use.

WILDCARD arguments may contain $* to designate any volume, * to designate any subvolume, and * to designate any file, in a syntax similar to: WILDCARD $*.DAT.*.

The wildcard indicator cannot be combined with letters.

Use WILDCARD instead of FILE whenever possible to reduce the number of possible Measure configurations. WILDCARD produces only one measurement entity to configure. FILE creates an entry for each item satisfying the wildcard. In addition, WILDCARD allows activity to be detected against files that have not yet been created.

To measure block-level activity against secondary partitions of files (for example, from a FUP LOAD), include those partitions explicitly.

You can filter out activity against selected files and types of files during the reporting phase. Specifying the most files that might be accessed during reporting in the data gathering phase allows more flexible reporting.

Syntax `WILDCARD <file set>`

Argument	Description
`<file set>`	Indicates a volume, subvolume, or file set.

Example

Valid WILDCARD Examples	Invalid WILDCARD Example
`WILDCARD $*.DAT.*` `WILDCARD $DATA3.SUBV.*`	`WILDCARD $DATA3.DAT.FL*` You cannot combine a volume, subvolume, or file name with a wildcard designator.

MEASRPT parameter reference

AVGCOMPRESSEDBYTES

Use AVGCOMPRESSEDBYTES to estimate the bytes needed per update record for bandwidth and disk storage. This figure approximates the maximum number of bytes likely to change in records or rows. For SQL, this is the number of columns specified in an update statement, plus the key.

Default 100 bytes

Syntax `AVGCOMPRESSEDBYTES <bytes>`

Argument	Description
`<bytes>`	The number of bytes needed per update record.

DEFAULTEXCLUDES

Use DEFAULTEXCLUDES to determine whether or not I/O resulting from OSIMAGE and ORSERV (FUP RELOAD) is excluded from the report.

Default `ON` (exclude results)

Syntax `DEFAULTEXCLUDES {ON | OFF}`

DURATION

Use DURATION to specify a length of time for collecting measurements in terms of minutes, hours, or days. For example, DURATION 48 HOURS, starts measurements at the time specified by the START parameter, and continues until 48 hours have elapsed.

Alternatively, you can use the STOP parameter to specify a time, such a 15:00:00, to stop measurements.

Default One day

Syntax `DURATION <num units> <unit>`

Argument	Description
`<num units>`	The amount of time for collecting measurements.
`<unit>`	The unit of time for collecting data. Enter one of: ◆ MINUTES ◆ HOURS ◆ DAYS

EXCLUDEALTKEYS

Use EXCLUDEALTKEYS to determine whether or not statistics for alternate key files are included in the report.

Default `ON` (exclude alternate keys)

Syntax `EXCLUDEALTKEYS {ON | OFF}`

EXCLUDEFILE

Use EXCLUDEFILE to determine whether or not statistics for a file set are included in the report.

Default Include all files gathered in the statistics phase

Syntax `EXCLUDEFILE <file set>`

Argument	Description
`<file set>`	The fully qualified file set to exclude. You can use wildcards.

Example `EXCLUDEFILE $DATA*.MYAPP*.FL?`

EXCLUDEPROGRAM

Use EXCLUDEPROGRAM to determine whether or not statistics for files updated by the specified programs are excluded from the report. For example, EXCLUDEPROGRAM $SYSTEM.SYS*.FUP excludes updates from FUP.

Default Exclude `FUP RELOADS`

Syntax `EXCLUDEPROGRAM <file set>`

Argument	Description
`<file set>`	The fully qualified program name to exclude.

EXCLUDETANDEMFILES

Use EXCLUDETANDEMFILES to determine whether or not statistics for files with file codes between 1 and 1000 are captured. Setting this parameter to ON results in the exclusion of object, EDIT and other files from the report.

Default Exclude files

Syntax `EXCLUDETANDEMFILE <$vol.subvol.fileset>`

Argument	Description
`<$vol.subvol.fileset>`	The fully qualified file set to exclude.

Example `EXCLUDETANDEMFILE $DATA*.MYAPP*.FL???`

FILEDETAIL

Use FILEDETAIL to determine whether or not the details of each file's activity are reported by period. If you do not specify FILEDETAIL, only summary totals are provided per period, along with a report of peak file activity.

Default `ON` (include file activity details)

Syntax `FILEDETAIL {ON | OFF}`

GETAUDITED

Use GETAUDITED to determine whether or not audited files are included in the report.

Default `ON` (include audited files)

Syntax `GETAUDITED {ON | OFF}`

GETENSCRIBE

Use GETENSCRIBE to determine whether or not Enscribe files are included in the report.

Default `ON` (include Enscribe files)

Syntax `GETENSCRIBE {ON | OFF}`

GETNONAUDITED

Use GETNONAUDITED to determine whether or not non-audited files are included in the report.

Default ON (include non-audited files)

Syntax `GETNONAUDITED {ON | OFF}`

GETSQL

Use GETSQL to determine whether or not NonStop SQL tables are included in the report.

Default ON (include SQL tables)

Syntax `GETSQL {ON | OFF}`

GETTMFDETAIL

GETTMFDETAIL determines whether or not TMF transaction information is reported on a per-program basis. If you want to report on TMF transaction information, the MEASFLS GETTMFDETAIL parameter must also specify ON.

Default ON

Syntax `GETTMFDETAIL {ON | OFF}`

INCLUDEFILE

Use INCLUDEFILE to determine whether or not statistics for a file set are included in the report. The default is to include all files gathered in the statistics phase.

If INCLUDEFILE is specified for one file set, then only files included with other INCLUDEFILE entries are output (unless excluded with EXCLUDEFILE).

Syntax `INCLUDEFILE <file set>`

Argument	Description
`<file set>`	The fully qualified file set to include.

Example `INCLUDEFILE $DATA*.MYAPP*.FL??`

INCLUDEPROGRAM

Use INCLUDEPROGRAM to determine whether or not statistics for files updated by the programs specified are included in the report. For example, INCLUDEPROGRAM $DATA2.APPLPROG.*, includes all programs in the $DATA2.APPLPROG file set.

If INCLUDEPROGRAM is specified for one set of programs, then only statistics explicitly named in other INCLUDEPROGRAM entries are output. To explicitly include a program, specify an argument similar to INCLUDEPROGRAM $DATA2.APPLPROG.<program name>.

If no INCLUDEPROGRAM entries are found, all program updates are included in the report unless excluded by the EXCLUDEPROGRAM parameter.

Default All program updates (except as excluded with EXCLUDEPROGRAM)

Syntax `INCLUDEPROGRAM <file set>`

Argument	Description
`<file set>`	The fully qualified program name to include.

INTERVAL

Use INTERVAL to specify a statistics reporting interval in terms of the amount of minutes, hours, or days that have elapsed. For example, INTERVAL 30 MINUTES, reports performance statistics every 30 minutes.

You can specify short reporting intervals for MEASFLS, then longer intervals for MEASRPT (using the MEASRPT INTERVAL parameter). Since short intervals can consume significant amounts of disk space and CPU, intervals of at least 30 minutes are recommended and should suffice for most installations. INTERVAL determines the level that peak activity can be reported against.

The MEASRPT INTERVAL parameter can exceed that previously specified by the MEASFLS INTERVAL parameter.

Default Interval specified by MEASFLS INTERVAL

Syntax `INTERVAL <num units> <unit>`

Argument	Description
`<num units>`	The amount of time for collecting data.
`<unit>`	The interval for collecting data. Enter one of: ◆ MINUTES ◆ HOUR ◆ DAY

LISTLIMIT

Use LIMITLIST to include only those files and programs in the report that generate the most output activity. You can limit the number of included files by specifying the <count> argument. For example, LISTLIMIT 20, lists only the 20 most active files and programs.

Syntax `LISTLIMIT <count>`

Argument	Description
`<count>`	Limits the number of reported files and programs to the first *n* files and programs specified as <count>.

MAXSTATPROGS

Use MAXSTATPROGS to change the default table size for the number of process entities that MEASRPT tracks.

Default 2000

Syntax `MAXSTATPROGS <num entries>`

Argument	Description
`<num entries>`	The number of entries to track.

MEASFH

Use MEASFH to identify the location of the MEASFH program.

When processing Measure data from a customer you often have to use the MEASFH object that matches the customer Measure date file. If specified, the MEASFH file name will be passed into MEASOPEN(); otherwise MEASOPEN() uses the default on the local system.

Syntax `MEASFH <file name>`

Argument	Description
`<file name>`	The name of an object file. The name may be partially qualified.

MEASFILES

Use MEASFILES to identify the locations of the file sets Measure will use to report statistics.

Syntax `MEASFILES <file set>.`

Argument	Description
`<file set>`	A fully qualified file set nameto be included in reports.

Example `MEASFILES $DATA3.MEASDAT.MDAT`

MAXSTATFILES

Use MAXSTATFILES to specify internal statistics arrays larger than 10,000 entries. Allocate one entry per file measured in the system.

Syntax `MAXSTATFILES <num files>`

Argument	Description
`<num files>`	The number of files.

MAXSTATPROGS

Use MAXSTATPROGS to change the default table size for the number of process entities that MEASRPT tracks.

Default 2000

Syntax `MAXSTATPROGS <num entries>`

Argument	Description
`<num entries>`	The number of entries

PROGDETAIL

Use PROGDETAIL to determine whether or not to report details about each program that updates the specified files.

Default `ON` (report details)

Syntax `PROGDETAIL {ON | OFF}`

PROGSTATS

Use PROGSTATS to determine whether or not to report statistics about programs that update the file set specified.

Default `ON` (report statistics)

Syntax `PROGSTATS {ON | OFF}`

REPORTRATE

Use REPORTRATE to determine whether or not statistics are reported as absolute numbers or on a per-second basis. For example, REPORTRATE ON, reports statistics on a per-second basis.

Default `OFF` (report absolute numbers)

Syntax `REPORTRATE {ON | OFF}`

RETENTION

Use RETENTION to estimate the maximum disk file capacity needed for an extended outage. For example, RETENTION 1 DAYS, specifies to retain one day.

Default One day

Syntax `RETENTION <num units> <unit>`

Argument	Description
`<num units>`	The amount of time for collecting data.

Argument	Description
<unit>	The interval for collecting data. Enter one of: ◆ HOURS ◆ DAYS

START

Use START to control when the reporting period starts. For example, START 2006-03-12 15:00:00, starts reporting at 1500 hours on March 12, 2006.

Syntax START <date> <time>

Argument	Description
<date>	The date in yyyy-mm-dd format. The default is the date at which the measurement was started by the MEASFL START parameter.
<time>	The time is hh:mm:ss format. The default is the time the measurement was started by the MEASFL START parameter.

STOP

Use STOP to control when the reporting period stops. For example, STOP 2006-03-12 15:00:00, stops reporting at 1500 hours on March 12, 2006.

Syntax STOP <date> <time>

Argument	Description
<date>	The date in yyyy-mm-dd format. The default is the date at which the measurement was started by the MEASFL START parameter.
<time>	The time is hh:mm:ss format. The default is the time the measurement was started by the MEASFL START parameter.

SUPPRESSZEROTOTALS

Use SUPPRESSZEROTOTALS to determine whether or not to report files for which statistics were captured but did not experience file updates.

Default ON (suppress zero totals)

Syntax SUPPRESSZEROTOTALS {ON | OFF}

CHAPTER 8

User Exit Functions

User exits allow you to 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 an unlimited number of specialized functions.

GoldenGate supplies:

- 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 or 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 GoldenGate checkpointing, 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.

Routine	Description
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_CHECKPOINT	Called just before an Extract or Replicat checkpoint is written.
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.

Exit Parameter	Description
ERR_INFO	This information is supplied with a discarded record to indicate the reason for the discard. The GoldenGate error number (ERR_NUM) is passed, along with a possible file or SQL error and error text.
FUNCTION_PARAM	Allows you to 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.
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.

Exit Parameter	Description
	◆ 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	Use when some of a target table's columns are present after mapping. Typically, use in conjunction with DECOMPRESS_RECORD.
DECOMPRESS_RECORD	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.

Function	Description
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.
GET_FILENAME	Retrieves the source or target file name.
GET_FOPEN_NUM	Gets the open number of the current file.
GET_NUM_COLUMNS	Returns the number of columns.
GET_RECORD	Use for custom field conversions that may not be handled by Extract or Replicat.
GET_RECORD_LENGTH	Retrieves the length of the target record.
GGS_REPORTMESSAGE	Allows a user exit to write to the report file.
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.

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.

Return value	Description
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 occured when attempting to fetch a record.
EXIT_FN_RET_INVALID_COLUMN	A non-existent column was referred to in the function call.
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 librabry 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

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.

Syntax

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)
05 compressed-rec   PIC X(4096).
05 compressed-len   PIC S9(4) COMP.
05 decompressed-rec PIC X(4096).
05 columns-present  PIC S9(4) COMP OCCURS 2000 TIMES.
05 source-or-target PIC S9(4) COMP.
05 result           PIC S9(4) COMP.
ENTER C "COMPRESS_RECORD" using compressed-rec, compressed-len,
         decompressed-rec, columns-present, source-or-target giving result.
```

Argument	Description
`columns_present` `columns-present`	An array indicating which columns to compress. For example, if the first, third and sixth columns are to becompressed, 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 based up to the maximum of 4096
`decompressed_rec` `decompressed-rec`	The record after it has been decompressed. The size is allocated by the user based up to the maximum of 4096
`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_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.

Syntax

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)
05 compressed-rec         PIC X(4096).
05 compressed-len         PIC S9(4) COMP.
05 decompressed-rec       PIC X(4096).
05 columns-present.
  10 columns-present-flag PIC S9(4) COMP OCCURS 2000 TIMES.
05 source-or-target       PIC S9(4) COMP.
05 result                 PIC S9(4) COMP.
ENTER C “DECOMPRESS_RECORD” using compressed-rec, compressed-len,
         decompressed-rec, columns-present, source-or-target giving result.
```

Argument	Description
`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 4096.
`decompressed_rec` `decompressed-rec`	The record returned in decompressed format. The size is allocated by the user up to the maximum of 4096
`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 audited 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.
```

Argument	Description
`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 audited 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.
```

Argument	Description
`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

C:

```
short  result;
short  keytag;
short  keyoff;
short  keylen;
result = GET_ALTKEY_INFO (&keytag, &keyoff, &keylen);
```

TAL:

```
?source usrdect
int result;
int .ext keytag;
int .ext keyoff;
int .ext keylen;
result := GET_ALTKEY_INFO(keytag, keyoff, keylen);
```

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.
```

Arguments	Description
`keytag`	The two-byte, generally alphnumeric, 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)
05 result            PIC S9(4) COMP.
05 col-idx           PIC S9(4) COMP.
05 col-name          PIC X(36).
05 source-or-target  PIC S9(4) COMP.
ENTER C "GET_COLUMN_INDEX" using col-idx, col-name, source_or_target,
                           giving result.
```

Argument	Description
`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)
05 result       PIC S9(4) COMP.
05 col-idx      PIC S9(4) COMP.
05 col-name     PIC x(36).
05 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.
```

Argument	Description
`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:

`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.
```

Argument	Description
`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)
05 EXIT-PARM PIC X(1000).
05 LEN       PIC s9(4) COMP.
05 RESULT    PIC s9(4) COMP.
ENTER C”GET_EXITPARAM_VALUE” using exit_parm, len giving result.
```

Argument	Description
`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 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.
```

Argument	Description
`extrba`	The current position in the GoldenGate trail.
`result`	A code indicating whether the call was successful or not.

GET_EXTSEQNO

Gets the sequence number for the current 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.
```

Argument	Description
`seqno`	The sequence number of the 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.
```

Argument	Description
`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.
```

Arguments	Description
`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)
05 num-columns      PIC S9(4) COMP.
05 source-or-target PIC S9(4) COMP.
05 result           PIC S9(4) COMP.
ENTER C "GET_NUM_COLUMNS" using num-columns, source-or-target
         giving result.
```

Argument	Description
`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

Data mapping from a source to a different target format is often required when extracting or replicating data. The COLMAP specification supported by both Replicat and Extract works sufficiently for most, but not all, 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 provides an ideal way to obtain information for the purpose of custom field conversions. For example, a proprietary date field (for example, YYDDD) in an Enscribe database could be converted to a standard SQL date in the target record by an exit while other columns would be mapped by Extract.

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.

```
short before_len;
short 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 manipute 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

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)
05 buf              PIC X(4096).
05 len              PIC S9(4) COMP.
05 io-type          PIC S9(4) COMP.
05 source-or-target PIC S9(4) COMP.
05 result           PIC S9(4) COMP.
ENTER C "GET_RECORD" using buf, len, io-type, source-or-target
                     giving result.
```

Argument	Description
`buf`	The record buffer for data returned by the Extract or Replicat programs. The memory for this buffer must be allocated by the user. The 4096 in the example is the maximum.
`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

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 to obtain this value when the mapped record is uncompressed. Set the EXIT-PARAMS structure member RECORD-LENGTH to this value.

Syntax

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)
05 record-length    PIC S9(4) COMP.
05 source-or-target PIC S9(4) COMP.
05 result           PIC S9(4) COMP.
ENTER C "GET_RECORD_LENGTH" using record-length, source-or-target
         giving result.
```

Argument	Description
`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.
```

Argument	Description
`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)
05 result    PIC S9(4) COMP.
05 trans-ind PIC S9(4) COMP.
ENTER C "GET_SOURCE_RECORD" using trans-ind giving result.
```

Argument	Description
`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.

Buffer

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
```

Argument	Description
`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 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.
```

Argument	Description
`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.

Argument	Description
`text`	The text of the message.
`result`	A code indicating whether the call was successful or not.

CHAPTER 9

Event Error and Warning Messages

GGS-EVT-STARTUP-PARAM —101

Cause Issued when an unrecognized or missing parameter is encountered during program startup.

Recovery 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.

Recovery 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 Issued when one of the following occurs:

- A Guardian file error was encountered during an operation on the file or process.
- Replicat has performed file-level operations such as RENAME, PURGE, ALTER, and other operations and cannot map the source file name to a target file name (the Guardian error code output is -2 or -3 in these cases).
- A FILE_CREATE_ operation has failed and a file attribute code is unrecognized by Replicat. The Guardian error output is 10000 plus the attribute code that failed.
- A replicated CONTROL operation attempted a position into a FORMAT 1 file beyond 4 gigabytes (error code -21).
- A problem occurred in positioning into a target file.

Recovery 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.

Recovery 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 GoldenGate Software.

Tokens GGS-TKN-AUDIT-FILE
GGS-TKN-AR-ERR

GGS-EVT-MEM-ALLOC —105

Cause Issued due to an internal problem allocating memory.

Recovery Restart the program. If the problem persists, contact 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
```

Recovery Note error message and contact 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.

Recovery 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.

Recovery 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 GoldenGate trail. View the message in the event file for more information.

Recovery 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.

Recovery 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 EXTTRAIL was reached during Extract processing.

Recovery Dependent on the situation. Possibilities include renaming or purging the oldest EXTTRAIL files, or increasing the MAXFILES option of the EXTTRAIL 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.

Recovery Restart the Extract program. If the problem persists, contact GoldenGate.

GGS-EVT-USER-EXIT—117

Cause A logic error probably exists when a user exit routine converts data to another format.

Recovery 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.

Recovery 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.

Recovery 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 GoldenGate is incompatible with the current operating system.

Recovery Install the correct version of GoldenGate.

GGS_EVT_MGR_CHILD_EXISTS—122

Cause The Manager child process was not stopped.

Recovery This is usually a warning message, but if this is a fatal 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.

Recovery 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 page 273.

Tokens GGS-TKN-RMT-HOST
GGS-TKN-TCP-PORT
GGS-TKN-TCP-ERR

GGS_EVT_SYNC_DUPPROC_ABEND— 160

Cause Syncfile abended.

Recovery 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.

Recovery 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.

Recovery 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.

Recovery Examine the previous error messages in the log in order to determine the exact cause.

GGS_EVT_SERVER_ABEND —193

Cause Collector abended.

Recovery 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.

Recovery 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.

Recovery Examine the error message to determine the exact nature of the error. Attempt to correct the error. Contact 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.

Recovery 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.

Recovery 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.

Recovery Contact 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.

Recovery 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.

Recovery 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.

Recovery 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.

Recovery 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.

Recovery 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.

Recovery 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.

Recovery 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.

Recovery 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.

Recovery 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 EXTTRAIL sequence and opened the next one.

Recovery 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.

Recovery 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.

Recovery 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.

Recovery 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 GoldenGate operations are continuing.

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

Recovery 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.

Recovery 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).
```

Recovery Warning only.

GGS_EVT_SYNC_DUP_FILE_ERR —253

Cause Syncfile failed to duplicate the file.

Recovery 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.

Recovery 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.

Recovery 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).

Recovery 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.

Recovery Waiting for a file event holds up Replicat processing. You can cancel the wait using the SEND REPLICAT <group>, 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.

Recovery 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.

Recovery None. Used to trace processing activity.

GGS-EVT-REP-STARTED—302

Cause The Replicat process started.

Recovery None. Used to trace processing activity.

GGS-EVT-EXT-STARTED—303

Cause The Extract process started.

Recovery None. Used to trace processing activity.

GGS-EVT-FILE-ERR-RECOVERED—304

Cause A file operation that previously encountered an error was retried successfully.

Recovery 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.

Recovery 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.

Recovery None. Used to trace processing activity.

GGS-EVT-EXT-STOP-USER—307

Cause Extract processing was terminated from GGSCI by a user.

Recovery None. Used to trace processing activity.

GGS-EVT-REP-STOP-NORMAL—308

Cause Replicat processing terminated normally.

Recovery None. Used to trace processing activity.

GGS-EVT-REP-STOP-USER—309

Cause Replicat processing was terminated from GGSCI by a user.

Recovery None. Used to trace processing activity.

GGS-EVT-MGR-PROCESS-RUNNING—310

Cause The Manager process is reporting that a process is alive.

Recovery 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.

Recovery 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.

Recovery Investigate the reason for the abend, fix the problem and restart the program.

GGS-EVT-MGR-STARTED—313

Cause The Manager process started.

Recovery 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.

Recovery 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.

Recovery 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.

Recovery 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.

Recovery 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.

Recovery 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.

Recovery 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 retryable reason.

Recovery 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 thelag for the process exceeds the configured threshold. This message warns that a problem may exist in the system.

Recovery 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.

Recovery None. Used to trace processing activity.

GGS-EVT-MGR-SLAVE —332

Cause A Manager reported error originating in the slave Manager process.

Recovery None. Used to trace processing activity.

GGS-EVT-MGR-CHILD —333

Cause A Manager reported error originating in a child process.

Recovery 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.

Recovery 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.

Recovery None. Used to trace processing activity.

GGS-EVT-NONSTOP —350

Cause Errors related to NonStop processing such as start backup, takeover checkpoint errors, etc.

Recovery None. Used to trace processing activity.

GGS_EVT_CPUUP —353

Cause Manager is reporting CPU recovery messages to EMS/log file for reference.

Recovery None. Information only.

GGS_EVT_SYNC_STARTED— 360

Cause Indicates Syncfile has started.

Recovery None. Informational only.

GGS_EVT_SYNC_STOP_USER —361

Cause Syncfile was stopped by the user, via GGSCI.

Recovery None. Information only.

GGS-EVT-COLL-OPENING-FILE—370

Cause Indicates that a Collector opened a file.

Recovery None. Used to trace processing activity.

GGS-EVT-COLL-CLOSING-FILE—371

Cause Indicates that a Collector closed a file.

Recovery None. Used to trace processing activity.

GGS_EVT_COLL_TERMINATING— 372

Cause Indicates that a Collector is terminating normally (due to Extract stopping).

Recovery 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.

Recovery 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.

Recovery 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.

Recovery None. Used to trace processing activity.

GGS-EVT-OPENED-SOURCE-FILE —382

Cause Issued by Extract when it opens a file that is not a GoldenGate trail or Logger trail file.

Recovery None. Used to trace processing activity.

GGS-EVT-CLOSED-SOURCE-FILE —383

Cause Issued by Extract when it closes a file that is not a GoldenGate trail.

Recovery 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.
```

Recovery 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.

Recovery None. Informational only.

GGS_EVT_CHKPT_REC_DELETED — 386

Cause Result from a GGSCI CLEANUP command.

Recovery None. Informational only.

GGS_EVT_EXTTRAIL-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.

Informational:

```
GGS INGO 387 Positioning for Begin <time> search, using
\NODE.$VOL.SUBVOL.AA000001.
```

Warning:

```
GGS WARNING 387 No data found in exttrail/logtrail for Begin <time>
search, using \NODE.$VOL.SUBVOL.AA00009.
```

XR-FILE-ERR—500

Cause This is an internal error code returned from the module that reads GoldenGate trails and does direct reads on user data files.

XR_LOGIC_ERROR —514

Cause Extract received an unexpected reply code from Replicat during direct load and generates the message:

```
Invalid reply code received.
```

Recovery 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.

Recovery Fix error and restart Logger. Resynchronization may be required.

GGS-EVT-LOGR-SHUTDOWN-RQST—1008

Cause Logger received a shutdown request from GGSCI.

Recovery None required. Informational.

GGS-EVT-LOGR-SHUTDOWN-COMPLETE—1009

Cause Logger completed a shutdown request from GGSCI.

Recovery None required. Informational.

GGS-EVT-LOGR-ABEND—1010

Cause Logger abended for reasons provided in another error message.

Recovery 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.

Recovery 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.

Recovery Examine the FILTER clauses in the Replicat parameter file. Also examine the REPERROR parameter.

GGS-EVT-POOL-CREATE-ERR—1500

Cause Memory allocation problems.

Recovery Contact GoldenGate Software.

GGS-EVT-LOGR-LOGCONF-ERR—1600

Cause The LOGCONF file may have been purged from configuration.

Recovery Contact GoldenGate Software.

GGS-EVT-LOGR-TRAIL-OPEN—1800

Cause Logger opened the next log file in the log trail.

Recovery None required. Informational.

GGS-EVT-LOGR-TRAIL-SWITCH—1801

Cause Logger will open the next log file in the log trail by request.

Recovery None required. Informational.

GGS-EVT-LOGR-LOGGING-DISABLED—1802

Cause Logger disabled logging by request.

Recovery None required. Informational.

GGS-EVT-LOGR-LOGGING-ENABLED—1803

Cause Logger enabled logging by request.

Recovery None required. Informational.

GGS-EVT-INTERCEPT-NOTBOUND—2000

Cause An application that opens a non-audited file for update access does not have GoldenGate's intercept library bound to it.

Recovery Bind GoldenGate's intercept library to the application.

ECONNABORTED— 4119

Cause A connection was closed by the internal software on your host machine.

Recovery 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.

Recovery 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.

Recovery 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.

Recovery 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.

Recovery 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.

Recovery 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.

Recovery 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.

Recovery 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.

Recovery 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

! command 87

! option, STOP REPLICAT 49

$SYSTEM.GGS.AUDCFG file 50

A

ABEND option
- -e 283
- FILEOPWARNING 174
- OPENWARNINGS 234
- REPERROR 243
- VERSIONERR 279

ABORTDUPERRWINDOW parameter 117

access mode, specifying 121

ACTIVE option, FILE 161

ADD COLUMN, replicating 247

ADD command
- ATCONFIG 64
- COORDINATOR 69
- EXTRACT 24
- EXTTRAIL 56
- LOGGER 50
- MARKER 76
- REMOTECHKPT 67
- REPLICAT 37
- RMTTRAIL 58

ADD DEFINE parameter 118

ADJUSTPRIORITY option, SEND LOGGER 53

after image
- compressing 160
- including or excluding 191
- indicator
  - *in ASCII output* 178
  - *returning* 299

AFTERFILTER option, SQLEXEC 170, 225

alias, for file name 169

ALLOCFILES parameter 119

ALLPARAMS option, SQLEXEC 170, 225, 227

ALLPROCESSES option
- AUTOSTART 129
- INFO ALL 91
- INFO EXTRACT 31
- INFO REPLICAT 42
- STATUS EXTRACT 36
- STATUS REPLICAT 48

ALTER command
- ATCONFIG 65
- COORDINATOR 70
- EXTRACT 28
- EXTTRAIL 59
- LOGGER 51
- REPLICAT 41
- RMTTRAIL 59
- SYNCFILE 85

ALTER operations, filtering 187

alternate key files
- creating 220
- purging 237

alternate process name
- Extract 27
- Replicat 40

ALTFILECHAR option, MAP 211

ALTFILERESOLVE parameter 120

ALTINPUT parameter 120

ALTLOC option, ADD ATCONFIG 64

ALTNAME option, FILE 162

ALWAYS option, DUP 141

APPEND option
DISCARDFILE 138
EXTFILE 154
REPSQLLOG 250
RMTFILE 258
ARCLOSECATALOG option, SEND EXTRACT 32
arithmetic operations, performing 293
array items in Enscribe definitions 152
ARRAYWIDTH option, EXPANDDDL 153, 321
ASC option, VIEW GGSEVT 94
ASCII output format 178
ASSUMETARGETDEFS parameter 122
ATEND option
STOP EXTRACT 37
STOP REPLICAT 48
AUDCFG option, INFO LOGGER 52
audit dumps, restoring 251
audit trails
auxiliary, bypassing 184
binary search of 156
disk level threshold 140, 273
gap in, ignoring 233
information, viewing 33, 69
management commands 64
markers, using 75
purging 240
reading on different CPU 124
audited files
including or excluding for extraction 160
Logger settings for 52
AUDITEND option, SEND EXTRACT 32
AUDITING parameter 123
AUDITREPS parameter 123
AUDITRETRYDELAY parameter 123
AUDRBA option
ADD EXTRACT 25
AUDSEQNO option
ADD EXTRACT 25
Audserv
binary search, performing 156
managing 34
parameters, passing 124
position, overriding 197
process names, assigning 126
program name, specifying 127
sequence names, prefix 126
AUDSERVCACHEBLOCKS parameter 123
AUDSERVCPU parameter 124
AUDSERVPARAM parameter 124
AUDSERVPREFIX parameter 126
AUDSERVPROCESS parameter 126
AUDSERVPROGRAM parameter 127
AUTORESTART parameter 127
AUTOSTART parameter 129
AUTOTRUNCATE option
FILE 162
MAP 168
RMTBATCH 254
auxiliary trails, see audit trails
AVGCOMPRESSEDBYTES parameter 332
AXCEL option, BIND PROGRAMS 80

B

-b parameter 282
BACKUPCPU option
ADD COORDINATOR 69
ADD EXTRACT 25
ADD REPLICAT 38, 39
ADD SYNCFILE 85
READER 240
BACKUPCPU parameter 130
BASE24 files, multiple 122
BASELIB stop delay 177
batch file, specifying 253
batch process, specifying 267
BCP options, FORMATASCII 179
before image
extracting 161
filtering 192
indicator, returning 299

BEFOREFILTER option, SQLEXEC 170, 225
BEGIN option
ADD EXTRACT 25
ADD REPLICAT 38
ALTER COORDINATOR 70
MACRO 209
BEGIN parameter 130
binary data
converting to number 308
preserving 291
BINARY function 291
BIND PROGRAMS command 79
BINDEXIT macro 28
BLKSIZE option, RMTBATCH 255
blocks
Collector, sizing 282
corrupt, skipping 149
small, grouping 265
BLOWFISH option
-ENCRYPT 284
RMTHOST 260
BOTH option, FILERESOLVE 175
BOUND option, INFO PROGRAMS 81
brackets, for array item 153
BREAK command 75
BRIEF option
INFO EXTRACT 30
INFO REPLICAT 42
buffer
Collector 282, 284
Extract 176
Logger 206
TCP 272
bulk I/O
enabling for unstructured data 131
excluding file-level operations from 187
updates
including or excluding 161
Logger settings for 52
-BULKIO parameter 282
-BULKIOLEN parameter 282
BULKIOLOAD parameter 131

C

C routines, calling 137
cache, SQL table definitions 123
calculations
artithmetic 293
dates 296
CAPTURE TABLEDEFS command 61
CASE function 291
century, changing 281
CHANGED option, DUP 141
CHANGELIB option
BIND PROGRAMS 80
LINK PROGRAMS 81
character strings, see *strings*
character substitution 245, 285
CHECKINTERVAL parameter 131
CHECKMINUTES parameter 131
CHECKPARAMS parameter 132
checkpoints
basing purges on 238
delay before writing 132, 229
deleting
Extract 29
Replicat 42
file location 46
flushing after 176
initial 24, 37
remote
adding 67
deleting 67
information about 67
remote file partitions 231
viewing
Extract 31
Replicat 43
CHECKPOINTSECS parameter 132
CHECKUNIQUEKEY parameter 132
CHILDSTATUS option, SEND MANAGER 22
CLEANUP command
EXTRACT 29
REPLICAT 41
CLEANUPSAVECOUNT parameter 133

CLEAREXCLUDELIST option, SEND EXTRACT 32
CLOSEACTION option, RMTBATCH 255, 286
CLOSEFILES option, SEND REPLICAT 44
closes, tracing 276
CLOSEWIN option, RMTBATCH 255
COBOL routines, calling 133
COBOLUSEREXIT parameter 133
CODE option, INFO FILES 63
COLHDRS option, FORMATASCII 179
Collector
alternate 145, 261
dynamic, specifying in parameter file 260
parameters 282
static, specifying in parameter file 259
collisions
handling after startup 251
resolving during processing 46, 193
COLMAP option
FILE 162, 166
MAP 211, 215
COLMATCH parameter 133
COLSTAT function 292
COLTEST function 292
columns
DDL changes, processing 189, 247
long names, managing 320
mapping 133, 166, 168, 215
net changes, extracting 188
redefined
including or excluding 320
omitting from Enscribe data 153
testing and converting 288
commands, GGSCI
by process name 75
executing from file 91
Extract, Replicat as unit 49
help for 90
history
saving 91
viewing 90
repeating
with modifications 88
without modification 87
scrolling 82
COMMENT
option, RMTBATCH 286
parameter 135
comments
in DDL creation script 324
in parameter files 135
in QSAM parameters 286
commit, within SQLEXEC statement 171, 226
COMPENSCRIBEMAPS option
FILE 218
MAP 211
COMPRESS option, RMTHOST 260
COMPRESS_RECORD function 346, 348
COMPRESSDELETES
option, FILE 162
parameter 135
compressed updates, including or excluding 184
compression, over TCP/IP 260
COMPRESSTHRESHOLD option, RMTHOST 260
COMPRESSUPDATES option, FILE 159
COMPUTE function 293
conditional statements
in file selection 164, 214
on column values 307
CONTINUE option
-e 283
VERSIONERR 279
CONTROL operations, filtering 187
conversion functions 288

Coordinator
adding 69
altering 70
deleting 71
information, viewing 71
name, specifying in parameter file 136
requests, sending to 72
starting 73
status, viewing 74
stopping 74
COORDINATOR parameter 136
COPY output format 181
corrupt blocks, skipping 149
CPU option
ADD COORDINATOR 69
ADD EXTRACT 25
ADD REPLICAT 38, 39
ADD SYNCFILE 85
READER 240
RMTHOST 260
CPU parameter 137
CREATE operations
alternate key files, replicating 220
filtering 185, 187
CREATES option, WAITFILEEVENT 279
CREATETEMPLATE option, MAP 211
CREATIONWINDOW option, WAITFILEEVENT 279
CUSEREXIT parameter 137

D

-d parameter 283
dash, for array item 153
data
binary, preserving 291
compressing 260
dividing into ranges 308
encrypting 146
formatting, location of 182
invalid, handling 245, 246, 285
output in external format 178, 181, 182, 253
processing in large blocks 156
selecting 163, 212, 266
see also *rows*
data definitions, see *DDL definitions*
data pump
adding 26
pass-through mode 235
data set, space allocation 286
data source, specifying 26
database, commands, GGSCI 61
DATE
function 293
option, FORMATASCII 179
DATEDIFF function 296
DATENOW function 296
dates
century, changing 281
converting 293
current, returning 296
differences, calculating 296
DBOP option, SQLEXEC 170, 225
DCBSN option, RMTBATCH 257
DDL changes, processing 189, 247
DDL definitions
cache space for 123
identical for source and target 122
lookups, bypassing 235
prompting for 321
source table, specifying 265, 283
SQL based on Enscribe 317
see also *Enscribe*
DDLGEN parameters 322
-DEBUG parameter 283
DEBUGONSTACKCHECK parameter 137
DECOMPRESS_RECORD function 348
DECRYPTTRAIL parameter 137
DEF option
ALTNAME 169
FILE 167
FILTER 164
MAP 211
WHERE 164
DEFAULT option
REPERROR 243
RETRYERR 252

default parameter settings, restoring 185
DEFAULTEXCLUDES parameter 332
DEFERREDRESOLUTION option, MAP 211
DEFGEN parameters 317
DEFINE statements, specifying 118
definitions, see *DDL definitions*
DEFSONLY option, INFO DDLDEFS 63
delay for
check for more data 123, 148
detecting new names 52
external event 279
Extract checkpoint 132
Extract read 265
GGSCI timeout 148
retrying failed operation 251
DELAY option, RETRYERR 252
DELAYSECS option
COORDINATOR 136
NETWORKCHECKPOINTS 231
DELETE command
ATCONFIG 65
COORDINATOR 71
EXTRACT 29
EXTTRAIL 59
LOGGER 51
REMOTECHKPT 67
REPLICAT 42
RMTTRAIL 60
SYNCFILE 86
deletes
compressing 135, 163
converting to inserts 198
converting to updates 277
filtering 186
omitting 161
DELIMITER option, FORMATASCII 179
DESC option
ADD COORDINATOR 69
ADD EXTRACT 25
ADD REPLICAT 38
ADD SYNCFILE 85
VIEW GGSEVT 94
description of group, adding
Extract 28
Replicat 40
DETAIL option
INFO ALL 91
INFO COORDINATOR 71
INFO EXTRACT 30
INFO REPLICAT 42
INFO SYNCFILE 86
STATUS COORDINATOR 74
STATUS EXTRACT 36
STATUS REPLICAT 48
DETECTLOCKS option, MAP 211
DICT option, INFO DDLDEFS 63
DICTIONARY
option, MAP 211
parameter 138
direct file extraction
starting at first record 236
syskey, specifying 270
discard file
aging 139
name, specifying 138
size, constraining 229
DISCARD option
-e 283
REPERROR 243
VERSIONERR 279
DISCARDFILE parameter 138
DISCARDROLLOVER parameter 139
DISKTHRESHOLD parameter 140
DISPLAYTRAILSWITCH parameter 140
distributed network transactions, monitoring 69
DOWN option
INFO EXTRACT 31
INFO REPLICAT 42
DOWNCRITICAL parameter 140
DOWNREPORT parameters 141
DUP
option, ADD ATCONFIG 64
parameter 141
DUPFILES option, ADD ATCONFIG 64

duplicates
file names, resolving 319
records, overriding 234
DUPONLYAFTEREVENT parameter 143
DUPPROCESS parameter 144
DURATION parameter 328, 332
DYNAMIC option, FILERESOLVE 175
dynamic ports
assignment delay 145
assignment list 144
DYNAMICPARTITIONS parameter 144
DYNAMICPORTLIST parameter 144
DYNAMICPORTREASSIGNDELAY parameter 145

E

-e parameter 283
EDIT PARAMS command 92
EMBEDDEDMACROS parameter 145
EMS
logging, using 285
messages to non-default Collector 145
EMSLOG parameter 145
ENCRYPT option, RMTHOST 260
-ENCRYPT parameter 284
ENCRYPT PASSWORD command 62
encryption
database user password 62
logon password 208
TCP/IP 260, 284
trail data 137, 146
ENCRYPTKEY option
ENCRYPT PASSWORD 62
LOGON 208
ENCRYPTTRAIL parameter 146
END parameter 146
end time, special run 146
Enscribe
data, formatting 152
DDL definitions, specifying
alternate file name 169
COLMAP clause 167, 216
FILTER clause 164, 213
WHERE clause 164, 214
dictionary
information about 63
specifying 138, 213
files
close threshold 233
creating on target 219
flushing at checkpoint 176
information about 61
mapping to SQL 317
timeout for open files 233
ENSCRIBE option, INFO FILES 63
entry-sequenced files
extracting from 26
replicating 147
using syskey 177
ENTRYSEQUPDATES parameter 147
ENV command 88
environment
referencing with
user exit functions 344
user tokens 172
values, returning 297
variables
setting 264
viewing 187
viewing 88
EOFDELAY parameter 148
ER commands 49
ERREPLYTIMEOUT parameter 148

error handling
BIND process 80
collisions 46, 193, 221
DDL anomolies 278
discard file, specifying 138
error 73 221
exceptions statement 223
format detection, Collector 283
GLOBALS entries 196
GoldenGate errors list 368
invalid data 245, 246, 285
last error, returning 298
missing target rows 199
NLD process 81
report rate, specifying 281
response, specifying 222, 242
retries, rules for 251, 252
SQLEXEC 172, 227
ERROR option, SQLEXEC 170, 225
ERROR59ROLLOVER parameter 149
ERRORS option
BIND PROGRAMS 80
LINK PROGRAMS 81
ETNEWFORMAT parameter 149
ETOLDFORMAT parameter 149
ETPURGE option, ALTER EXTRACT 29
ETROLLOVER option, ALTER EXTRACT 29
EVAL function 296
EVENT
option, DUP 141
parameter 149
event log, viewing 93
EXACTKEY option, ENTRYSEQUPDATES 147
EXCEPTION option, REPERROR 243
exceptions
handling 242
map statement, creating 223
EXCEPTIONSONLY option, MAP 211
EXCLUDE option, DUP 141
EXCLUDEALTKEYS parameter 333
EXCLUDEFILE parameter 150, 333
EXCLUDEFILECODE option, DUP 141
EXCLUDEGGSTRANSRECS parameter 151
EXCLUDELASTDIGIT option, INFO FILES 63
EXCLUDEPROGRAM parameter 333
EXCLUDESUFFIXCHAR parameter 152
EXCLUDESYSTEM parameter 317
EXCLUDETANDEMFILES parameter 334
EXCLUSIVE option, ALTINPUT 121
EXCLUSIVEOPEN option, ENTRYSEQUPDATES 147
EXEC option
SQLEXEC 170, 225
WAITFILEEVENT 279
EXIT-CALL-RESULT function 341
EXIT-CALL-TYPE function 341
EXITMAP option, FILE 162
EXITPARAM option
FILE 162
MAP 211
EXIT-PARAMS function 342
EXIT-REC-BUF function 344
EXPAND option, RETRYERR 252
EXPANDDDL parameter 152, 317
EXPANDGROUPARRAYS option, EXPANDDDL 319
EXTENTS option
ADD EXTTRAIL 56
DISCARDFILE 138
EXTFILE 154
LOG 203
RMTFILE 258
EXTFILE parameter 153
EXTFILESOURCE option, SPECIALRUN 267
EXTRACOLS option, FORMATASCII 179

Extract
alter messages, suppressing 269
buffer threshold 176, 272
lag, viewing 31
logon, specifying 207
read byte size, changing 156
requests, sending 32
starting
automatically 127, 129
manually 36
status, viewing 36
stopping 31, 37
see also *Extract group*
extract file
as data source 267
specifying in parameter file 153
Extract group
adding 24
altering 28
deleting 29
history, deleting 29, 133
information, viewing 30
parameter file, creating 92
specifying in parameter file 155
EXTRACT parameter 155
extract trail, see *trail*
extraction
from
entry-sequenced files 26
log trail 26
source file 26, 266
trail files 26
online, specifying 24
start time 130
EXTRBA option
ADD REPLICAT 38
ALTER COORDINATOR 70
EXTRBA option, ADD EXTRACT 27
EXTSEQNO option
ADD REPLICAT 38
ALTER COORDINATOR 70
EXTSEQNO option, ADD EXTRACT 27
EXTTRAIL option
ADD REPLICAT 38
ALTER COORDINATOR 70
DELETE COORDINATOR 71
EXTTRAIL parameter 155
EXTTRAILSOURCE option
SPECIALRUN 267

F

-f parameter 284
FASTIO parameter 156
FASTPOSITION parameter 156
FASTREADS
option, ALTINPUT 121
parameter 156
FASTUNLOAD option, SOURCEISFILE 266
FASTUNLOADSHARED option, SOURCEISFILE 266
FC command 88
FETCH_CURRENT_RECORD function 349
FETCH_CURRENT_RECORD_WITH_LOCK function 350
FETCHCOMPS parameter 156
FETCHLASTIMAGE parameter 156
field conversion functions 288
fields, see *columns*
file numbers, hard coded 205
FILE option
FORMATASCII 179
FORMATSQL 183
FILE parameter
Extract 162
Logger 157
MEASFLS 328
FILEAGEDAYS parameter 173
FILEDETAIL parameter 334
FILEEXCLUDE parameter 174
file-level operations, including or excluding 187
FILEMEGABYTES parameter 329
FILEOP option, REPERROR 243
FILEOPWARNING parameter 174
FILERESOLVE parameter 175
files, BASE24, see *BASE24 files*

files, database
as data source 266
creating on target 219
excluding from
extraction 271
Logger 150
replication 228
wildcards 174
logging 157
mapping
source to target 210
to SQL from Enscribe 317
missing 174
names
alternate 168
duplicate 319
resolving over TCP/IP 195
returning 299
substituting strings for 169
opening as shared 284
partitioned 144
specifying for extraction 162
statistics, generating with MEASFLS 325
tracing 276
see also *Enscribe*
files, entry-sequenced, see *entry-sequenced files*
files, QSAM 285
files, unstructured, see *unstructured files*
FILLSHORTRECS parameter 176
filter clause
in extraction rules 163
in Logger rules 158
in mapping rules 213
in relation to query 171, 226
FILTER option
FILE 162
MAP 211
FILTERVIEW parameter 176
FLUSHALWAYS option, ENTRYSEQUPDATES 147
FLUSHCHECKPOINT parameter 176
FLUSHCSECS parameter 176
FLUSHSTATS option, SEND LOGGER 53
FORCEBIND option, BIND PROGRAMS 80
FORCECOMMIT option, SEND COORDINATOR 72
FORCESTOP option
SEND EXTRACT 32
SEND REPLICAT 44
FORCESTOPDELAY parameter 177
FORCEUSESYSKEY parameter 177
FORMATASCII parameter 178
FORMATLOAD parameter 181
FORMATLOCAL parameter 182
FORMATSQL parameter 182
formatting activities, location of 182
FTYPE option, RMTBATCH 255
functions, conversion
memory management 183
syntax and usage 288
FUNCTIONSTACKSIZE parameter 183
FUP messages, displaying 278
FUPOPTIONS option, DUP 141

G

generation data sets, creating 257
GET_COLUMN_NAME function 353
GET_ENV_VALUE function 354
GET_EXITPARAM_VALUE function 356
GET_EXTRBA function 356
GET_EXTSEQNO function 357
GET_FILENAME function 358
GET_NUM_COLUMNS function 359
GET_RECORD function 360
GET_RECORD_LENGTH function 362
GET_SYSKEY_LENGTH function 363
GET_TRANSACTION_IND function 364
GETALTFILES option, AUDSERVPARAM 125
GETALTKEYS parameter 183
GETAPPLOPS parameter 184
GETAREXCLUDELIST option, SEND EXTRACT 32
GETARFILELIST option, SEND EXTRACT 32
GETARFILESTATS option, SEND EXTRACT 32
GETARPARAMS option, SEND EXTRACT 32
GETARPROCESS option, SEND EXTRACT 32
GETARSTATS option, SEND EXTRACT 32

GETAUDITED
option, FILE 160
parameter 334
GETAUDITED option, DUP 141
GETAUXTRAILS parameter 184
GETBEFOREUPDATES option, FILE 161
GETBULKIO option, FILE 161
GETCOMPS parameter 184
GETCREATES parameter 185
GETDEFAULTS parameter 185
GETDELETES parameter 186
GETENSCRIBE parameter 334
GETENV
function 297
parameter 187
GETEXTARSTATS option, SEND EXTRACT 32
GETFILEOPS parameter 187
GETGROUPTRANSOPS option, SEND REPLICAT 44
GETINSERTS parameter 188
GETMARKERS parameter 188
GETMAXTRANSOPS option, SEND REPLICAT 44
GETNETCHANGES parameter 188
GETNETWORKCHECKPOINTS option, SEND REPLICAT 44
GETNEWCOLUMNS parameter 189
GETNONAUDITED parameter 334
GETNONDATACHANGES option, AUDSERVPARAM 125
GETPARTONLYPURGEDATAS parameter 189
GETPURGEDATAS
option, AUDSERVPARAM 125
parameter 190
GETPURGES parameter 190
GETREADERINFO option, SEND COORDINATOR 72
GETREPLICATES parameter 190
GETROLLBACKS
option, SEND EXTRACT 32
parameter 191
GETSQL parameter 335
GETSTATS option, SEND LOGGER 53
GETSYSTEMUTILS parameter 329
GETTCPSTATS option, SEND EXTRACT 32
GETTMFDETAIL parameter 329, 335
GETTRANSINFO option
SEND COORDINATOR 72
SEND EXTRACT 32
GETUNSTRUCTURED option, FILE 161
GETUPDATEAFTERS parameter 191
GETUPDATEBEFORES parameter 192
GETUPDATES parameter 192
GETVAL function 305
GGENVIRONMENT option, @GETENV 299
GGFILEHEADER option, @GETENV 300
GGHEADER option, @GETENV 299
GGS option
-ENCRYPT 284
RMTHOST 260
GGS_EMSMESSAGE function 366
GGS_REPORTMESSAGE function 367
GGSCI
commands
logging of, controlling 207
session, saving 91
timeout, setting 148
GGSDLL option, LINK PROGRAMS 81
GGSLIB
as Guardian User Library 79
binding to application 79
configuring 50
delay for Logger response 207
information about 52
programs, information about 80
GGSLIB option, BIND PROGRAMS 80
GGSPARM.LOGPARM file 50
GGSSRL option, LINK PROGRAMS 81
GGSSRL, linking to application programs 81
GGSSUBVOL option, HOST 195
GLOBALS parameters, invalid 196
Golden Gate Software Command Interface, see *GGSCI*
group
old name format 232
see also *Extract group* or *Replicat group*
GROUP option, RMTTASK 262
GROUPCMD option, ADD MARKER 76
groups, see *Extract group* or*Replicat group*

GROUPTRANSOPS parameter 192
Guardian User Library 79

H

HANDLECOLLISIONS
option, MAP 211, 221
option, SEND REPLICAT 44
parameter 193, 251
header, record
user tokens, defining 172
HEARTBEAT parameter 194
heartbeat record, issuing 194
HELP command 90
HIDEGAPS option, ENTRYSEQUPDATES 147
high value, constraining 306
HIGHVAL function 306
history
cleanup rule 133
deleting
Extract 29
Replicat 41
GGSCI commands 90
target transactions 218
HISTORY command 90
home terminal
messages, handling 195
specifying 194, 284
-HOMETERM parameter 284
HOMETERM parameter 194
HOMETERMMESSAGES parameter 195
HOST parameter 195
host, see *remote host*
HOTSWAP option, SEND LOGGER 53

I

ID option, SQLEXEC 170, 225
IF function 307
IGNORE option
FILEOPWARNING 174
FILTER 164
OPENWARNINGS 234
IGNORE option, REPERROR 243
IGNOREALTFILES option, AUDSERVPARAM 125
IGNOREALTKEYS parameter 183
IGNOREAPPLOPS parameter 184
IGNOREAUDITED option, DUP 141
IGNOREAUDITED option, FILE 160
IGNOREAUXTRAILS parameter 184
IGNOREBEFOREUPDATES option, FILE 161
IGNOREBULKIO option, FILE 161
IGNORECOMPS parameter 184
IGNORECREATES parameter 185
IGNOREDELETES parameter 186
IGNOREFILEOPS parameter 187
IGNOREINSERTS parameter 188
IGNOREMARKERS parameter 188
IGNORENETCHANGES parameter 188
IGNORENEWCOLUMNS parameter 189
IGNOREPARAMERROR parameter 196
IGNOREPARTONLYPURGEDATAS parameter 189
IGNOREPURGEDATAS
option, AUDSERVPARAM 125
parameter 190
IGNOREPURGES parameter 190
IGNOREREPLICATES parameter 190
IGNOREROLLBACKS
option, SEND EXTRACT 32
parameter 191
IGNORETMFDUMPS parameter 196
IGNOREUNSTRUCTURED option, FILE 161
IGNOREUPDATEAFTERS parameter 191
IGNOREUPDATEBEFORES parameter 192
IGNOREUPDATES parameter 192
IMMEDIATE option, FILERESOLVE 175
INCLUDE parameter 196
INCLUDECOMMENTS parameter 324
INCLUDEFILE parameter 335
INCLUDEFILECODE option, DUP 141
INCLUDENOTNULL parameter 323
INCLUDENULL parameter 323
INCLUDEPROGRAM parameter 335

INCLUDEREDEFS
option, EXPANDDDL 153, 320
parameter 323
INCLUDESYSTEM parameter 317
indexes, appending to fields 319
INFO command
ALL 90
ATCONFIG 66
COORDINATOR 71
DDLDEFS 63
ER 49
EXTRACT 30
EXTTRAIL 60
FILES 63
LOGGER 52
MARKER 78
PROGRAMS 80
REMOTECHKPT 67
REPLICAT 42
RMTTRAIL 60
SYNCFILE 86
INFO MANAGER command 22
initial load
errors, resolving 193
Extract group, adding 26
purging target files before 168
Replicat group, adding 40
INSERTALLRECORDS
option, MAP 211
parameter 197
INSERTDELETES parameter 198
INSERTMISSINGUPDATES parameter 199
inserts
applying all records as 197, 218
converting to updates 277
creating for missing target rows 199
creating from updates 199
filtering 188
omitting 161
INSERTUPDATES parameter 199
INTERVAL
option, LAGSTATS 201
parameter 329
IP address, alternate 261
IPINTERFACE parameter 199

J

JTSOFFSET option, ALTINPUT 121
Julian timestamp, formatting 295

K

-k parameter 284
key, encryption 208, 261, 284
key, primary or unique
alternate
file creates, handling 183
specifying 220
checking on target 132
matching for entry-sequenced records 147
substitute for 218
KEYCOLS option
FILE 162
MAP 211
TABLE 219
KEYNAME option, RMTHOST 261
-KEYNAME parameter 284
KILL command
ER 49
EXTRACT 31
SYNCFILE 86

L

-l parameter 284
lag
check interval 201
function for returning 298
report interval 200
reporting 43
all processes 90
based on time threshold 31
Extract 31, 35
Extract and Replicat 204
on schedule 201
Replicat 43, 45

LAG command
EXTRACT 31
REPLICAT 43
LAG option
@GETENV 298
INFO EXTRACT 31
INFO REPLICAT 42
LAGCRITICALparameters 200
LAGINFO parameters 200
LAGOFF option
SEND EXTRACT 32
SEND REPLICAT 44
LAGREPORT parameters 201
LAGREPORTOFF option
SEND EXTRACT 32
SEND REPLICAT 44
LAGREPORTON option
SEND EXTRACT 32
SEND REPLICAT 44
LAGSNAPSHOT option
SEND EXTRACT 32
SEND REPLICAT 44
LAGSTATS
option, SEND EXTRACT 32
option, SEND REPLICAT 44
parameter 201
LASTERR option, @GETENV 298
LIKE option, RMTBATCH 257
LIMITLIST parameter 336
LIMITRECS parameter 202
LINK PROGRAMS command 81
LIST parameter 202
literal string, passing to user exit 169, 219
load utilities, formatting data for 178
locks
ignoring 241
reducing number of 230
log
Collector 284
event 93
LOG command 91
log mode, viewing 52
LOG parameter 203
LOG STOP command 91
log table, purging 237
log trail, Logger
as data source 26
creating 51
deleting 51
location, viewing 52
markers, using 75
rolling over 54
specifying 203
see also *Logger*
LOGFILEOPENS parameter 205
LOGFILESBEHIND parameters 204
Logger
adding and configuring 50
BASELIB delay to read queue 177
buffer flush point 206
configuration files, deleting 51
CPU, specifying 137
file numbers, adjusting 205
GGSCI stop, enforcing 231
information, viewing 52
logging rules, specifying 150, 157
maximum messages 241
opens and closes, specifying 205
parameters, altering 51
pausing 161
requests, sending 53
statistics, viewing 277
stopping 55
system priority 236
timeout 207
LOGGER option, ADD MARKER 76
LOGGERFILENUM parameter 205
LOGGERFLUSH parameters 206
LOGGERFLUSHRECS parameter 206
LOGGERTIMEOUTSECS parameter 207
LOGGGSCICOMMANDS parameter 207
logging, suspending 161
logical name, for a query 170, 225
LOGINFO option, SEND LOGGER 53
LOGON parameter 207

logon, specifying 207
LOGPARM file 50
LOGTRAIL option, ADD REPLICAT 38
low value, constraining 306
LOWVAL function 306

M

-m parameter 284
MACRO parameter 208
MACROCHAR parameter 210
macros
- alternate character, defining 210
- creating 208
- expanding in quoted string 145
- including libraries 196
- listing in process report 202
- TACL 143

Manager
- autorestart feature 127
- autostart feature 129, 251
- backup CPU 130
- excluding from configuration 210
- history cleanup specification 133
- information, viewing 22
- maintenance frequency 131
- parameter file
  - *creating* 92
  - *executing* 22
- port
  - *dynamic list* 144
  - *TCP/IP* 236
- retry count 228
- starting 23
- status, viewing 23
- stopping 24

MANAGERREQUIRED parameter 210
MAP
- option, SQLEXEC 171, 226
- parameter 210

MAPEXCEPTION option, MAP 211
MAPEXCLUDE parameter 228
MAPID option, MAP 211
mapping
- columns 133, 215
- Enscribe to SQL 317
- secondary partitions 234
- source and target files 210
- user tokens 316

markers
- filtering 188
- viewing 78

MAXABENDRESTARTS parameter 228
MAXCOLNAMELEN option, EXPANDDDL 320
MAXDISCARDRECS parameter 229
MAXETCHECKPOINTSECS parameter 229
MAXFILES option
- ADD EXTTRAIL 56
- EXTFILE 154
- RMTFILE 258

MAXFILESPERMEAS parameter 330
MAXRETRIES option
- COORDINATOR 136
- NETWORKCHECKPOINTS 231
- RETRYERR 252

MAXSTATFILES parameter 337
MAXSTATPROGS parameter 337, 338
MAXTRANSMEM parameter 229
MAXTRANSOPS parameter 230
MAXVARCHARLEN option, SQLEXEC 170, 225
MAXWILDCARDENTRIES parameter 230
MEASFH parameter 337
MEASFILES parameter 330, 337
MEASFLS parameters 328–332
MEASRPT parameters 332–339
Measure statistics, using with GoldenGate 325
MEGABYTES option
- ADD EXTTRAIL 56
- ADD RMTTRAIL 58
- DISCARDFILE 138
- EXTFILE 154
- LOG 203
- RMTFILE 258

memory, managing
file name allocations 119, 232
GoldenGate functions 183
transaction storage 229
wildcard entries 230
MGRPORT option, RMTHOST 260
MINKEEP options, PURGEOLDEXTRACTS 238
missing files, handling 168, 174

N

NAME option, DUP 141
NAME option, RMTHOST 261
NAMES option
COLMATCH 133
FORMATASCII 180
NETWORKCHECKPOINTS parameter 231
NEW option, -e 283
NO DUP option, ADD ATCONFIG 64
NO DUPFILES option, ADD ATCONFIG 64
NO PURGE option, ADD ATCONFIG 64
NOADJUSTPRIORITY option, SEND LOGGER 53
NOALTFILERESOLVE parameter 120
NOAUDITREPS parameter 123
NOCHECKUNIQUEKEY parameter 132
NOCOLMAP option
FILE 167
MAP 217
NOCOMPENSCRIBEMAPS option, MAP 211
NOCOMPRESS option, RMTHOST 260
NOCOMPRESSDELETES parameter 135
NOCOMPRESSUPDATES option, FILE 159
NODENUM option, HOST 196
NODISKTHRESHOLD parameter 140
NODISPLAYTRAILSWITCH parameter 140
NOENCRYPTTRAIL parameter 146
NOEXACTKEY option, ENTRYSEQUPDATES 147
NOEXPANSION option, EXPANDDDL 319
NOFASTPOSITION parameter 156
NOFASTREADS parameter 156
NOFETCHCOMPS parameter 157
NOFETCHLASTIMAGE parameter 157
NOFILTERVIEW parameter 176
NOFIXLONGNAMES option, EXPANDDDL 321
NOFLUSHALWAYS option, ENTRYSEQUPDATES 147
NOFLUSHCHECKPOINT parameter 176
NOFORCEUSESYSKEY parameter 177
NOFORMATASCII parameter 178
NOGETNONDATACHANGES option, AUDSERVPARAM 125
NOHANDLECOLLISIONS
option, MAP 211
option, SEND REPLICAT 44
parameter 193
NOHDRFIELDS option, FORMATASCII 180
NOHIDEGAPS option, ENTRYSEQUPDATES 147
NOINITAUXPOSITION parameter 197
NOLIBBIND option, BIND PROGRAMS 80
NOLIST parameter 202
NONAMES option
FORMATASCII 180
FORMATSQL 183
non-audited files, capturing 79, 81
NONE option, RMTHOST 260
NONOINITAUXPOSITION parameter 197
NONTMF option, INFO FILES 63
NOOVERRIDEUPS parameter 234
NOPARAMS option, SQLEXEC 170, 225
NOPASSTHRU parameter 235
NOPURGEDATAALTFILE parameter 237
NOQUOTE option, FORMATASCII 180
NOREADTHRULOCKS parameter 241
NOREPNEWCOLUMNS parameter 247
NORESOLVEDUPS option, EXPANDDDL 319
NORESTARTCOLLISIONS parameter 251
NORESTORE parameter 251
NOSUPPRESSALTERMESSAGES parameter 269
NOSYSKEY option, FORMATASCII 180
NOTCPSOURCETIMER parameter 273
NOTHRESHOLD parameter 273
NOTRANSTMTS option, FORMATASCII 180
NOTSTOPPABLE parameter 231
NOUPDATEDELETES parameter 277
NOUPDATEINSERTS parameter 277
NOWAITNEXTMODIFIED option, ALTINPUT 121

NOY2KCENTURYADJUSTMENT parameter 281
NULLISSPACE option, FORMATASCII 180
nulls, including with DDLGEN 323
numbers
converting to character 313
invalid, substituting 246, 285
NUMBIN function 308
NUMEXTRACTS parameter 231
NUMFILES
option, LOG 203
parameter 232
NUMSTR function 308

O

-o SHARED parameter 284
OBEY
command 91
parameter 232
obey file, specifying 91
occurs in Enscribe DDL 152
OLD option, -e 283
OLDGROUPNAMING parameter 232
OMITAUDITGAPCHECK parameter 233
OMITCOMMENTS parameter 324
OMITDELETES option, FILE 161
OMITINSERTS option, FILE 161
OMITNOTNULL parameter 323
OMITNULL parameter 323
OMITREDEFS
option, EXPANDDDL 153, 320
parameter 323
OMITUPDATES option, FILE 161
ON options, FILTER 164
ONCE option, SQLEXEC 171, 226
ONEFILE option, ALTINPUT 121
opens
access mode 121
number permitted 205
tracing 276
warning message 233
OPENTIMEOUT
option, ALTINPUT 121
parameters 233
OPENWARNINGS parameter 233
operations
grouping 46
Replicat
grouped, statistics for 46
ignoring or including 184
type, returning 300
ORACLE option, FORMATSQL 183
Oracle SQL*Loader 180
OUTOFSYNC option, -e 283
OVERRIDEDUPS parameter 234
OWNER option
ADD EXTTRAIL 56
DISCARDFILE 138
EXTFILE 154
LOG 203
RMTFILE 258

P

-p parameter 284
PARAMBUFSIZE option, SQLEXEC 170, 225
PARAMCHECK parameter 235
parameter files
alternate name
Extract 27
Replicat 40
Syncfile 270
comments in 135
editing 92
linking with OBEY 232
location
CHGNOTE 101
Extract 101
GLOBALS 95
Manager 97
Replicat 109
verifying 235
viewing 92

parameters, GoldenGate
defaults, restoring 185
invalid, ignoring 196
verifying 132
parameters, in procedure or query 305
parameters, Logger
altering 51
GGSLIB intercept library 50
viewing 52
parameters, SQLEXEC
managing 172, 227
specifying 171, 226
PARAMS option
ADD COORDINATOR 69
ADD EXTRACT 25
ADD LOGGER 50
ADD REPLICAT 38
ADD SYNCFILE 85
ALTER LOGGER 51
BIND PROGRAMS 80
LINK PROGRAMS 81
RMTBATCH 254
RMTHOST 261
RMTTASK 262
SQLEXEC 170, 225
PARTIALCOLSOK option, MAP 211
partitions
dynamic 144
mapping 234
selecting for extraction 165
PARTITIONS option, FILE 162
PARTMAP
option, MAP 211
PARTMAP parameter 234
PASSTHRU parameter 235
password, encrypting 62, 208
pause, in logging 161
PLACEHOLDERS option, FORMATASCII 180
PORT
command 236
option, RMTHOST 259
parameter 236
port number
alternate remote host 262
dynamic
assignment list 144
reassignment delay 145
Manager 236
TCP/IP 284
POSITIONFIRSTRECORD parameter 236
PREFIX option, COLMATCH 133
PRI option
ADD COORDINATOR 69
ADD EXTRACT 25
ADD REPLICAT 38, 39
ADD SYNCFILE 85
READER 240
RMTHOST 261
priority
Extract 27
Logger 52, 54
Replicat 39
PRIORITY parameter 236
PROCESS option
ADD COORDINATOR 69
ADD EXTRACT 25
ADD REPLICAT 38
ADD SYNCFILE 85
EXCLUDEFILE 151
FILE 159
LOG 203
READER 240
SEND LOGGER 53
-PROCESS parameter 282
process stack, debugging 137
processes
all, controlling 49
automatic startup 127, 129
information on all 90
managing from single point 92
reports for
diagnostics and management 82
running processes 277
stopped processes 140, 141
restarting 117, 228

PROCESSINFO option, SEND LOGGER 53
PROGDETAIL parameter 338
PROGRAM option
ADD COORDINATOR 69
ADD EXTRACT 25, 28
ADD REPLICAT 38
ADD SYNCFILE 85
EXCLUDEFILE 151
FILE 159
INFO COORDINATOR 71
INFO EXTRACT 31
INFO REPLICAT 42
READER 240
programs
binding to GGSLIB and GGSSRL 79
transactions, ignoring 184
PROGSTATS parameter 338
PROTECTED option, ALTINPUT 121
PROTECTEDOPEN option, ENTRYSEQUPDATES 147
PURGE operations, filtering 187, 189, 190
PURGE option
ADD ATCONFIG 64
DISCARDFILE 138
REPSQLLOG 250
RMTFILE 258
PURGEDATA operations
during initial load 168
filtering 187, 190
PURGEDATAALTFILES parameter 237
PURGEMEASFILES parameter 330
PURGEOLDEXTRACTS parameter 237
PURGEOLDTASKS parameter 239
PURGERESTORE parameter 240
PURGES option, WAITFILEEVENT 279

Q

QSAM parameters
configuration 285
RMTBATCH 256
query
executing 169, 224
extracting values from 305
QUERY option, SQLEXEC 170, 225

R

-R parameter 285
RANGE function 165, 308
RANGE option
ALTINPUT 121
FILE 162
MAP 211
ranges, assigning 122, 165, 215, 308
READER parameter 240
Reader process, creating 240
READTHRULOCKS parameter 241
RECEIVEQWARN parameter 241
RECLEN option, RMTBATCH 255
record length, returning 300
RECORDNAMEPROMPTING parameter 321
records, trail
count, reporting 248
header, user tokens 172
padding at end 176
preceding with process information 276
records, transaction
confirming all processed 33
count, reporting 248
ignoring all but Replicat 184
RECSONLY option, INFO DDLDEFS 63
refresh interval, specifying 275
REFRESH MANAGER command 22
REFRESH option, SEND LOGGER 53
REFRESHTMFINFO command 67
remote file, specifying 253, 258
remote host
alternate 261
heartbeat marker, sending 194
identifying 195
specifying 259
tasks on 262
remote trail, see *trail*
RENAME operations, filtering 187, 190
RENAMEBUMPDELAY parameter 241
RENAMEDELAY option, FILE 161

RENAMES option, WAITFILEEVENT 279
renames, accounting for 161
REPERROR
option, MAP 211
parameter 242
REPLACEBADCHAR parameter 245
REPLACEBADNUM parameter 246
Replicat
commands within markers 78
file operation messages, suppressing 269
lag, viewing 43, 45
logon, specifying 207
report file, alternate 250
requests, sending 44
SQL, viewing 224
starting
automatically 127, 129
manually 47
status, viewing 48
stopping 48
transactions
framing within TMF 123
grouping 192
ignoring 190
retry delay 251
splitting into smaller sizes 230
see also *Replicat group*
Replicat group
adding 37
altering 41
deleting 42
history, deleting 41, 133
information, viewing 42
parameter file, creating 92
specifying in parameter file 246
REPLICAT parameter 246
REPNEWCOLUMNS parameter 247
report files
aging 249
alternate name
Extract 27
Replicat 40
SQL log 250
debugging 249
extents, specifying 248
rolling over 34, 45
see also *reports*
REPORT HANDLECOLLISIONS option, SEND EXTRACT 44
REPORT option
ADD COORDINATOR 69
ADD EXTRACT 25
ADD REPLICAT 38
ADD SYNCFILE 85
BIND PROGRAMS 80
LAGSTATS 201
LINK PROGRAMS 81
SEND EXTRACT 32
SEND REPLICAT 44
REPORT parameter 247
REPORTCOUNT parameter 248
REPORTFILEEXTENTS parameter 248
REPORTRATE parameter 338
REPORTROLLOVER parameter 249
reports
Extract processing 33
normal process termination 140
on demand 82
processes that are stopped 141
Replicat processing 45
Syncfile 85
viewing 84
write interval, specifying 247
see also *report files*
REPORTTMFEXCEPTIONS parameter 249
REPSQLLOG parameter 250
RESET option
COLMATCH 133
REPERROR 244
RESETARSTATS option, SEND EXTRACT 32
RESETEXTARSTATS option, SEND EXTRACT 32

RESETMINUTES option, AUTORESTART 128
RESETTCPSTATS option, SEND EXTRACT 32
RESOLVEDUPALPHAINDEX option, EXPANDDDL 319
RESOLVEDUPFULLNAME option, EXPANDDDL 319
RESOLVEDUPGROUP option, EXPANDDDL 319
RESOLVEDUPINDEX option, EXPANDDDL 319
RESTARTCOLLISIONS parameter 251
RESTARTINTERVAL parameter 251
RESTORE parameter 251
restricting Manager's interface 199
RESULT function 345
RETENTION parameter 338
RETRIES option, AUTORESTART 128
RETRYDELAY parameter 251
RETRYERR parameter 252
RETRYOP option, REPERROR 244
RETRYOPEN option, REPERROR 244
RMTBATCH parameter 253, 285
RMTFILE parameter 258
RMTHOST parameter 259
RMTHOSTALT parameter 261
RMTTASK parameter 262
RMTTRAIL parameter 263
rollbacks, filtering 36, 191
ROLLOVER
- option, DISCARDFILE 138
- option, SEND EXTRACT 32
- option, SEND LOGGER 53
- parameter 263

ROLLREPORT option
- SEND EXTRACT 32
- SEND REPLICAT 44

rows
- duplicate, overriding 234
- full image, extracting 156
- length, matching
  - *to source* 217
  - *to target* 167
- locked, ignoring 241
- log table, deleting 237
- number extracted, limiting 202
- selecting 163, 212
- see also *data*

RUNTIME option, END 146

S

SAVE option
- CLEANUP EXTRACT 29
- CLEANUP REPLICAT 41

scrolling commands 82
SECURE option
- DISCARDFILE 138
- EXTFILE 154
- LOG 203
- RMTFILE 258

security
- logon 207
- password encryption 62, 208
- TCP/IP encryption 260, 284
- trail access 58
- trail files 146

segment file, GGSLIB
- altering 51
- creating 50
- information about 52

SELECTVIEW option, SOURCEISFILE 266
SEND command
- COORDINATOR 72
- ER 49
- EXTRACT 32
- LOGGER 53
- PROCESS 75
- REPLICAT 44
- REPORT 82

separators, defining in definitions file 318

SETENV parameter 264
SETGROUPTRANSOPS option, SEND REPLICAT 44
SETMAXTRANSOPS option, SEND REPLICAT 44
SETMODE
filtering 187
issued by Replicat 221
SHARED option, ALTINPUT 121
shell script, on file close 286
SHORTREADDELAY parameter 265
SHOWCH option
INFO COORDINATOR 71
INFO EXTRACT 31
INFO REPLICAT 42
SHOWLOGGED option, INFO LOGGER 52
SHOWSYNTAX option, MAP 211
source definitions file, specifying 265
source files, see *files, database*
SOURCEDEFS parameter 265
SOURCEISFILE parameter 266
SOURCEISTABLE
option, ADD EXTRACT 26
parameter 266
SOURCEROW option, SQLEXEC 172, 227
SPACE option, RMTBATCH 256, 286
spaces, trimming
leading 312
leading and trailing 315
trailing 314
SPECIALRUN
option, ADD REPLICAT 38
parameter 267
SQL
errors, see *error handling*
log, alternate 250
output format 182
Replicat, viewing 224
views, filtering 176
SQL DML output 182
SQL option, INFO FILES 63
SQL Server BCP copy program 179
SQL*Loader 180
SQLCI LOAD records, generating 181
SQLEXEC option
FILE 162
MAP 211
SQLLOADER option, FORMATASCII 180
SQLNAME option
FILE 162
MAP 211
START command
COORDINATOR 73
ER 49
EXTRACT 36
LOGGER 55
MANAGER 23
REPLICAT 47
SYNCFILE 86
START parameter 330, 339
start time
initial checkpoint
Extract 26
Replicat 39
special run 130
statistics
Audserv 34
current transaction 82
Extract and Replicat 82
interim 34
lag
Extract 35, 201
Replicat 45, 201
Logger 54, 277
Replicat 45
reporting 42
TCP/IP 34
STATOPTIONS parameter 268
status
all processes 90
Extract 30, 36
Logger 52
Manager 23
Replicat 42, 48
Syncfile processes 87

STATUS command
AUDITTRAIL 66
COORDINATOR 74
ER 49
EXTRACT 36
LOGGER 55
MANAGER 23
REPLICAT 48
SYNCFILE 87
STATUS option
SEND COORDINATOR 72
SEND EXTRACT 32
SEND REPLICAT 44
STOP command
COORDINATOR 74
ER 49
EXTRACT 37
LOGGER 55
MANAGER 24
REPLICAT 48
SYNCFILE 87
STOP option
SEND COORDINATOR 72
SEND EXTRACT 32
SEND REPLICAT 44
STOP parameter 331, 339
stop time, special run 146
STOPDELAYCSECS parameter 177
STRCAT function 310
STRCMP function 310
STREQ function 310
STREXT function 311
STRFIND function 311
strings
comparing 310, 313, 316
concatenating 310, 312
converting
binary to number 308
character to number 308
number to character 313
to uppercase 315
length, returning 312
literal, passing to user exit 169, 219
portion of, extracting 311
position in, determining 311
spaces in, trimming 312, 314, 315
substituting
for table name 169
one for another 314
STRLEN function 312
STRLTRIM function 312
STRNCAT function 312
STRNCMP function 313
STRNUM function 313
STRRTRIM function 314
STRSUB function 314
STRTRIM function 315
STRUP function 315
SUFFIX option, COLMATCH 133
SUPPRESSALTERMESSAGES parameter 269
SUPPRESSFILEOPMESSAGES parameter 269
SUPPRESSMARKERMESSAGES parameter 269
SUPPRESSZEROTOTALS parameter 339
SUSPENDED option, FILE 161
swap volume, specifying 270
SWAPVOL parameter 270

Syncfile
debugging 275
events
altering execution time 143
defining 149
interval 131
files, selecting 141
process for duplicating files 144
restarts, specifying 117
timing of duplication processes 143
see also *Syncfile group*
Syncfile group
adding 84
altering 85
deleting 86
information about 86
parameter file, alternate name 270
starting 86
status, viewing 87
stopping 86, 87
SYNCFILE parameter 270
syntax
command, editing 88
parameter, editing 92
Replicat, viewing 224
syskey
format, specifying 270
using on entry-sequenced tables 177
SYSKEYCONVERT parameter 270
system
changing current 92
command on file close 286
home terminal
changing 194
messages, handling 195
local files, information about 63
names, excluding from definitions file 317
parameters, setting 123
priority for Logger 236
SYSTEM command 92

T

-t parameter 285
TABLE parameter, see *FILE parameter* 271
TABLEEXCLUDE parameter 271
tables, see *files, database*
TACL
commands, issuing from GoldenGate 77
macro, using for duplication 143
messages, displaying 278
TACLCMD option
ADD MARKER 76
DUP 141
TAL routines, calling 272
TALUSEREXIT parameter 272
TANDEMFILES option, INFO FILES 63
TANDEMTSOFFSET option, ALTINPUT 121
tape dumps, restoring 251
TARGET option
FILE 162
MAP 216
target system, see *remote host*
TARGETBLKSIZE option, RMTBATCH 256, 286
TARGETDEF option
FILE 162
MAP 211
TARGETDICT option, MAP 211
TARGETFTYPE option, RMTBATCH 256, 286
TARGETNAME option, FILE 162
TARGETRECLEN option, RMTBATCH 256, 286
tasks
specifying 26, 40, 262
status, viewing 36
viewing 31
TASKS option
INFO ALL 91
INFO EXTRACT 31
INFO REPLICAT 42
STATUS EXTRACT 36
STATUS REPLICAT 48

TCP/IP
Manager port 236
process name 272, 282
retries before switching destination 273
statistics 34, 285
timestamps, adjusting 273
tracing 285
TCPFLUSHBYTES parameter 272
TCPIPPROCESSNAME parameter 272
TCPIPSWITCHERRS parameter 273
TCPSOURCETIMER parameter 273
-TCPSTATS parameter 285
TEMPLATE option, ALTINPUT 121
template, for target file creation 219
tests
conditional 307
presence of column 292
value selection 291, 296
text
in markers 77
in parameter file 135
THRESHOLD
option, LAGSTATS 201
parameter 273
TIME option
FORMATASCII 179
VIEW GGSEVT 94
timestamp
adjusting 241, 273
end, batch processing 146
transaction, returning 299
TLFKEY option, GETENV 305
TMF
dumps
ignoring 196
information about 68
refresh interval, setting 68
exceptions, handling 249, 274
in Measure statistics 329
see also *audit trails*
TMF option, INFO FILES 63
TMF refresh interval 275
TMFDUMPAGE
command 272
parameter 274
TMFDUMPAGE command 68
TMFDUMPINFO command 68
TMFDUMPTABLEENTRIES
command 68
parameter 274
TMFEXCEPTIONS parameter 274
TMFREFRESHINTERVAL
command 68
parameter 275
TMFTRAILINFO command 69
TMFTRAILTRACE parameter 275
TOKEN function 316
TOKENS option, FILE 162
tokens, user
retrieving 316
specifying 172
TRACE option, SQLEXEC 170, 225
TRACE parameter 275
TRACEALLOPENS parameter 276
TRACECLOSES parameter 276
TRACEOPENS parameter 276
TRACEPROCESSIOS parameter 276
TRACESTATS parameter 277
tracing
file user activity 276
SQLEXEC activity 172, 227

trail
adding and configuring 56, 58
altering 59
as data source 26, 267
decrypting 137
deleting 59, 60
encrypting 146
files
maximum number 231
purging 237
rolling over 35
information, viewing 60
old format, using 149
rolling to new file 263
specifying in parameter file 155, 240, 263
start point in 39
switch messages, suppressing 140
see also *records, trail*
trail, Logger, see *log trail, Logger*
TRANSABORT option, REPERROR 244
transaction indicator, returning 300
TRANSACTION option, SQLEXEC 171, 226
transactions
grouping 192
ignoring
all but Replicat 184
Replicat 190
splitting into smaller sizes 230
see also *error handling*
TS option, FORMATASCII 179

U

UNBOUND option, INFO PROGRAMS 81
underscore, for array item 153
UNIT option, RMTBATCH 257
UNMAPPEDALTFILECREATES option, MAP 211
UNSTRUCT option, INFO FILES 63
unstructured files
including or omitting 161
Logger settings for 52
UP option
INFO EXTRACT 31
INFO REPLICAT 42
UPDATEDELETES parameter 277
UPDATEINSERTS parameter 277
updates
applying based on source 199
before images, including 161
compressed
including or excluding 184
Logger settings for 52
compressing 159, 218
converting to inserts 199
filtering 192
full image, extracting 156
omitting 161
updates, bulk I/O 52
upper case, converting to 315
UPREPORT parameters 277
USEALTKEY option, MAP 211
USEBRACKETS option, EXPANDDDL 318
USEBYTEPOSITION option, SYSKEYCONVERT 271
USECHECKPOINTS option, PURGEOLDEXTRACTS 238
USEDASH option, EXPANDDDL 318
USEDEFAULTS option, MAP 217
USEKEY option, SOURCEISFILE 266
USENEXTMODIFIED option, ALTINPUT 121
User exit libraries 346
C function prototypes 346
C structures and defines 346
COBOL working storage 346
TAL function prototypes 346
TAL literals and values 346
user exits
calling 133, 137, 272
demo code 340
for column mapping 168
multiple 28, 41
parameters 340
passing literals to 169, 219
user library, binding 79
USER option
EXCLUDEFILE 151
FILE 159
user tokens, see *tokens*
USERECORDNUMBER option, SYSKEYCONVERT 271

USESAMENULLS parameter 324
USESOURCERECLENGTH option, MAP 211
USESTOPSTATUS argument 239
USETARGETDEFLENGTH option, FILE 162
USETWOUNDERSCORES option, EXPANDDDL 318
USEUNDERSCORE option, EXPANDDDL 318

V

-v parameter 285
VALONEOF function 316
VERBOSE parameter 278
VERSIONERR parameter 278
VIEW command
GGSEVT 93
PARAMS 92
REPORT 84
views, capturing 176
volume
remote node 195
swap 270
VOLUME option, RMTBATCH 256, 286

W

WAIT option
STOP EXTRACT 37
STOP REPLICAT 48
WAITFILEEVENT parameter 279
WAITMINUTES option, AUTORESTART 128
WAITNEXTRBA option, ALTINPUT 121
WAKE command 75
WARN option
FILEOPWARNING 174
OPENWARNINGS 233
VERSIONERR 279
WARNAFTER option, WAITFILEEVENT 279
WARNRATE parameter 281
where condition, in file selection 164, 214
WHERE option
FILE 162
MAP 211
WILDCARD parameter 331
wildcards
excluding files from 174, 228
memory, managing 230
resolving 120
rules. specifying 175
using 158

X

-x parameter 285

Y

Y2KCENTURYADJUSTMENT parameter 281

Z

ZEROFILL option, EXPANDDDL 153